2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics

An Efficient Algorithm for Robot Maze-Solving

Hongshe Dang, Jinguo Song, Qin Guo
School of Electric and Information Engineering, Shaanxi University of Science and Technology
Xi'an City, Shaanxi, 710021, China
E-mail: songjinguo_123@yahoo.com.cn

***Abstract*—this paper presents an efficient IEEE standard robot maze-solving algorithm. According to the actual situation of the robot searching maze, this algorithm improved the flood fill algorithm in maze-solving. Comparing with the results of using flood-fill algorithm directly, experiments show that this algorithm works better and more efficiently, and also, it has the advantage of little searching time and high speed of maze-solving. So it can be used to some areas like robot finding path.**

Keywords-robot; maze-solving; finding path; intelligent algorithm

I. INTRODUCTION

Today, robots are used in more projects as substitution for human. In most of these projects, robots should move or walk and find their path. Therefore, the ability of robot to consciously find their way around the terrain is needed in human life. At the present time, a maze-solving robot, self-contained without using energy source, is more necessary than it was in previous years. The speed of robot to find its path, affected by the applied algorithm, acts the main part in the present projects. In this paper the proposed algorithm is used to increase the solving speed and its efficiency. A lot of maze-solving robotic competitions are held around the world to achieve faster and superior robots. To test the new algorithm, it is used in one of these competitions, called "Micromouse" and the algorithm is tested in this kind of competition's maze. While the rules and the maze of this competition is formulated by the IEEE community, and this robot's competition is called IEEE Standard Miromouse Competition today. The IEEE standard maze has 16×16 square cells, and each cell's length and width are both 18 centimeters. Micromouse is a small wheeled-robot that consisting infrared sensors, motors and controller and act. It can find the optimal path to get the destination automatically in the IEEE standard maze. Generally, the destination cell is center cell in 16×16 maze. The start cell is the left down corner cell or right down corner cell. This is very important. About this competition's rule and the IEEE standard maze is introduced in [1]. In this paper, after overview the flood fill algorithm in section 2, the problem of flood fill algorithm is introduced in section 3. And then, in section 4 a new proposed algorithm is presented and in section 5 the results are stated. Finally, conclusion is given in section 6.

II. OVERVIEW OF FLOOD FILL ALGORITHM

Flood Fill algorithm is one of the best maze solving algorithms. The flood fill algorithm involves assigning values to each of the cells in a maze where these values represent the distance from any cell on a maze to the destination cell. The flood fill algorithm mainly contains four parts: update walls, flood maze, turn determination and move to next cell. The four parts of this algorithm is introduced in [2].

III. PROBLEM OF THE FLOOD FILL ALGORITHM

Although the robot using flood-fill algorithm can find the shortest path to destination, but some problems still exist when using this flood fill algorithm in the process of maze-solving. The problems are shown as follows.

- First, in each cell the robot arrived, it must run the four steps: update walls, flood maze, turn determination and move to next cell. That's the problem. Time is the key factor in this competition. This will affect the robot searching speed extremely.
- Second, flooding one time of a maze with 256 cells will costs the robot much time.
- Third, in order to find the destination as soon as possible, which direction the robot should to make when it gets to a cross cell which has two or more directions have the same value.

In this situation, a new algorithm is presented. It works better and solves the above problems. The detail is shown in the following parts.

IV. THE NEW PROPOSED ALGORITHM

According to the actual maze-solving, it is found that a lot of small channels exist in a maze. Three channels are shown out in Fig. 1 and other existed channels are not shown out. The channel means that once the robot enter, it only can move to its front cell (here suppose the robot's front has the high priority). So, in order to reduce the robot's calculation, there is no need to perform the four steps in those channels when it comes to searching the maze.

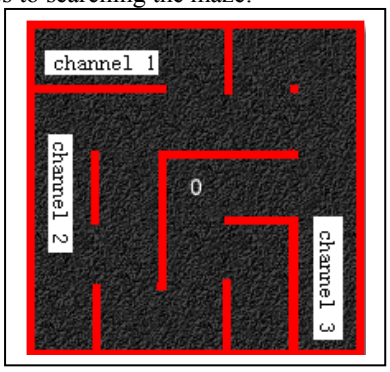

Figure 1. channel explanation

978-0-7695-4151-8/10 $26.00 © 2010 IEEE
DOI 10.1109/IHMSC.2010.119

And also, the robot do not need to flood the whole maze until getting to a next cross cell which has at least two open directions.

Based on this situation, we presented the new algorithm. The whole flow of the new algorithm is as shown in Fig. 2.This new algorithm mainly has five parts which is shown as follows.

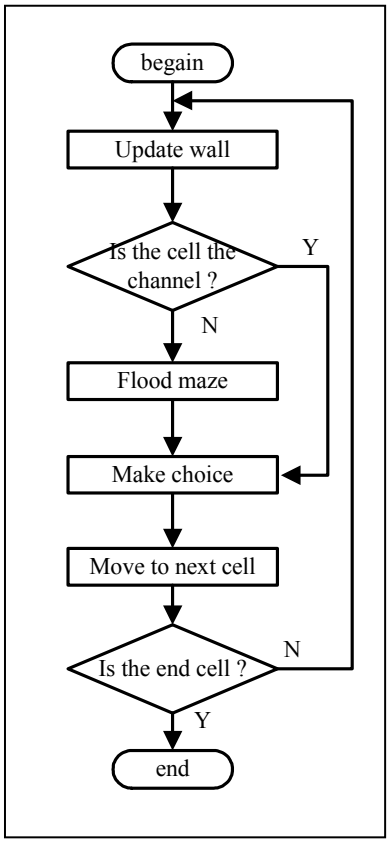

Figure 2. the new algorithm flow chart

A. Update the Wall

According this new algorithm, we assume the robot only can turn four directions in each cell through building the four directions' coordinate system. when the robot stays at a new cell, the first thing it should do is to detect its right, left, front and back, checking whether there has a wall in that direction. The detecting method is using the infrared sensor which is installed on the robot. The location of five installed infrared sensors on the robot is as shown in Fig. 3. Actually, it only needs to detect the right, left and front direction. Because it comes from its back, so this direction always has no wall. While these walls will affect the distance values in the maze, so we need to keep track of them [2]. There are 256 cells in a real maze, so 256 bytes will be more sufficient to keep track of the walls. There are 8 bits in the byte for a cell.

The first 4 bits can represent the walls leaving you with another 4 bits for your own use. It is initialized with 0. If the south direction has a wall, the S bit is set to 1; if the north direction has a wall, then the N bit is set to 1 and the same situation for W, E bit. A typical cell byte is shown in table 1.

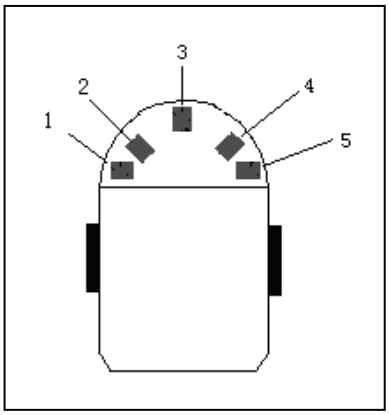

Figure 3. the robot's infrared sensor location 1,2 the left and right sensor,3,the front sensor;2,4 the left 45 degree sensor and right 45 degree sensor.

TABLE I. THE WALL INFORMATION .W-WEST;S-SOUTH;E-EAST;N-NORTH

Bit No.	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Wall					W	S	E	N

The four directions (W, S, E, and N) is the absolute direction. It's the direction of maze, so it's not changed once fixed; the robot's direction is the relative direction, its reference is the robot itself. So it changes a lot while the robot searching maze. The wall information detected by the robot is relative, so the robot must change the relative direction's wall information to absolute direction. Because comparing with the relative direction, the wall information is easier to process and save under the absolute direction. The robot's relative direction is shown in Fig. 4. Number 0 presents the front; number 1 is right; number 2 is the back; number 3 is left.

We define 0 as the North of absolute direction; 1 as the East; 2 as the South and 3 as the West and *RDIR* as the current relative direction, *ADIR* as the absolute direction. The *RDIR* is changed often when the robot making turn action. At the begin, $RDIR = 0$, if robot turns right, the *RDIR* add 1; if robot turns back, RDIR add 2,if the $RDIR > 3$,then the $RDIR = RDIR\%4$; if the robot turns left, *RDIR* subtract 1, if $RDIR < 0$,then the $RDIR = |RDIR|\%4$.

In order to change the relative direction to absolute direction, the converting table is showing in table 2.

Using this table the robot can easily change its direction, and then save the wall information to the right bit. Firstly, we define absolute North as the front of the robot's relative direction when it begins to start. This situation the $RDIR = ADIR$ .The *RDIR* records the robot's current relative direction, so if the $RDIR = 3$, that means the robot current relative front is absolute west. That is the $ADIR = 3$, according to this, we can judge out robot's right is the absolute north; its back is the east; its left is the south. So it can easily save the wall information to absolute direction using this.

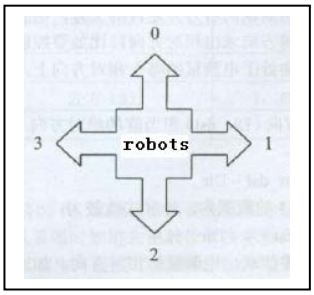

Figure 4. the robot's relative direction

TABLE II. THE CONVERSION TABLE

RDIR	ADIR
Robot's front	ADIR
Robot's right	(ADIR +1) %4
Robot's back	(ADIR +2) %4
Robot's left	(ADIR +3) %4

Remember that every interior wall is shared by two cells, so when updating the one cell's wall information, you must update its neighbor cell as well.

B. Judge Whether the Cell is a Channel Cell

The judging method is using the walling information. If the robot finds both have walls on its right and left side, this means that the robot itself stays at a channel cell. Otherwise, it's not.

If the robot is in a channel, there is no need to flood the whole maze, only need to go forward until finding a cross cell. While the cell is not a channel cell, the new algorithm is implemented as the flood fill algorithm. While the robot enters a channel without an exit in maze, it just needs to back to its previous cross cell to make other choice.

C. Flood Maze

Flood maze is also the key section in this algorithm. It implements the same as the flood fill algorithm. The details are described in [2].

D. Make Choice

After flooding the maze, the robot only needs to turn to the direction which has the smallest distance values. It's easy to make choice in this situation according to the flooding distance values. But there exists another situation we must consider. Due to the maze destination is the center of maze cell and the maze is random. So, there may be appear two or more directions have the same distance values. In this case, a rule is needed to process this situation. Here, because the robot turning action will cost time and limit its speed, so moving front has the highest priority, and moving back has lowest priority. The right and left action is random, which is no fixed.

When the robot needs to take action, it must convert the absolute direction to relative direction so that it can know which action it should make. We first calculate the $CDir$ using (1). According to table 3, the robot can find out which direction it should turn. For example, if the robot in a cell with $RDIR = 2$ and the east of maze has an exit according the wall information and distance values, that means the $dest\_DIR = 1$, so the $CDir = 1$ according to (1). Therefore, the robot must turn left. That's the right action the robot should make.

$$CDir = (dest\_DIR + 4 - RDIR)\%4 \quad (1)$$

TABLE III. THE INVERSE CONVERSION TABLE

CDir	Turn direction
0	robot's front
1	robot's right
2	robot's back
3	robot's left

E. Move to Next Cell

Once known the turn direction, the robot only needs to make turn action and move one cell's distance to the next cell and repeat this algorithm until finding the destination cell.

V. ANLYSIS AND COMPAREATIONS

In order to compare the two algorithms, simulation is carried out by using the two different algorithms to solve a maze separately. The final time results are shown in the following Tables. We suppose the flood-fill algorithm as the A1, the new algorithm as the A2, their maze-solving time are shown in the Table 4.

TABLE IV. THE EXHAUSTED TIME TABLE (S: SECOND)

Time Table	8×8 maze	16×16 maze
A1 (s)	37.4	311.2
A2 (s)	34.0	104.0

Simulation is also carried out between the Deep-first maze-solving algorithm and the new proposed algorithm. The maze-solving time are shown in the form 2. We suppose the Deep-first algorithm as the A3, the new modified algorithm as the A2, their final maze-solving time are shown in the Table 5.

TABLE V. THE EXHAUSTED TIME TABLE (S: SECOND)

Time Table	8×8maze	16×16maze
A3(s)	50.4	251.1
A2(s)	30.9	83.0

From the Table 4 and Table 5, the results show that the new proposed algorithm is better. its searching time is much shorter than the time of the Deep-first algorithm and the flood fill algorithm. And also the new algorithm has the advantage of high efficiency of maze-solving and can finding the shortest path only searching parts of the maze.

In Chinese 2009 IEEE Standard Micromouse Competition, the semifinal maze and the final maze are shown in Fig. 5 and Fig. 6. In semifinal competition, one robot used the new algorithm. Its searching route is shown by the real line in Fig. 5, and exhausted time is 109 seconds, its last spurting route is shown by the broken red line in Fig.5,

and the exhausted time is 21 seconds. In final competition, using the same algorithm, the same one robot searching route is shown by the real line in Fig. 6, and exhausted time is 97 seconds, its last spurting route is shown by the broken line in Fig. 5, and the exhausted time is 22seconds.and also is the fastest robot.

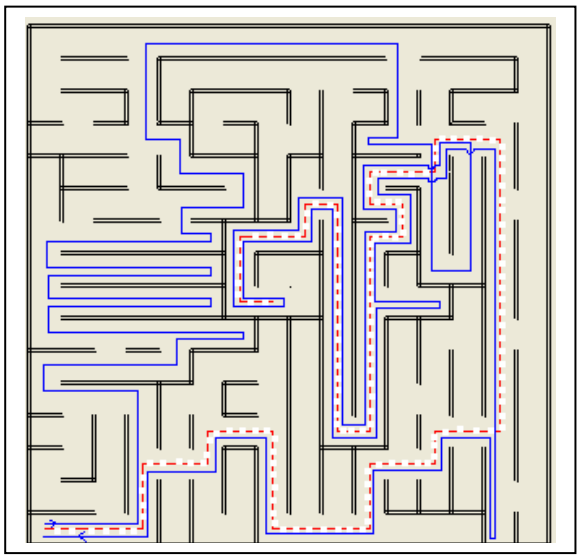

Figure 5. the route of the semifinal maze

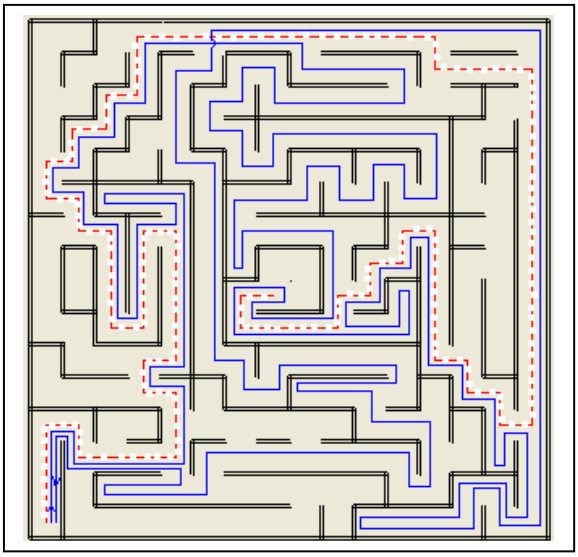

Figure 6. the route of final maze

Above all, there always have a lot of channels in a maze, so this new proposed algorithm decreases the robot's operation time largely and therefore, increasing its searching speed.

VI. CONCLUSION

Maze-solving involves Control Engineering and Artificial Intelligence. Using a good algorithm can achieve the high efficiency of finding the shortest path. The proposed maze-solving algorithm works better and has short searching time and low space complexity, and it is significant for robot's finding path in some areas like maze-solving.

ACKNOWLEDGMENT

This paper is supported by projects of the Study on Digital Broadcast and Television Controller of Shaanxi province Grant#2008K05-26 and the Graduate Innovation Fund of Shaanxi University of Science and Technology**.** We would like to thank the supports sincerely.

REFERENCES

[1] http://ewh.ieee.org/reg/1/sac/Main/RegionalEvents/StudentConf/MicromouseRules.pdf.

[2] Kazerouni B.H.Moradi."Variable Priorities in Maze-Solving Algorithms for Robot's Movement",Seville:Proceedings IEEE International Conference on Industrial Informatics. vol. 6,pp181-185, 2003.

[3] Tondra De, Dfew Hall. "The Inception of Chedda: A detailed design and analysis of Micromouse",University of Nevada, Las Vegas Senior Design. vol. 5, pp39-44, 2004.

[4] Huo Zhe Fu. "A Design and Realize for Micromouse's Maze-solving",Microcomputer Appfications.vol 9,pp59-62. 2008.

[5] Jin Liang Fang, Yi Jun Zhou. "Micormouse Based on ARM of IEEE Standard", Machine Building and Automation.vol.5,pp99-101,2008.

[6] UK Micromouse contest, held each year in United Kingdom.

[7] Kazuo Sugihara , John Smith. "Genetical algorithms for adaptive motion planning of anautonomous mobile robots" ,Problems IEEE Trans.USA:SIM,19997.

[8] MicroMOuse, California State University at Northridge, http//homepage.mac.com/SBenkovic/MicroMouse/index.html.

[9] Dieguez A R,Sanz R,Lopez J. "Deliberative on-line local path plannning for autonomous mobile robot",Journal of Intelligent and Robotic Systems,vol. 37,pp1-19,2003.

[10] http://www.micromouseonline.com.

[11] MicroMousel02 IEEE standard Micromouse data manualm, 2007.

[12] LV Qiang WANG Ke-ke."The Research of Range Measurement with Infrared Sensor in MicroMouse",Micocomputer Information.vol. 12,pp118-120, 2008.

[13] Chatila R."Path Planning and Environment Learning in a Mobile Robot System", In:Proc of the European Conf on Al,1982.

[14] Frank Lingelbach."Path Planning using Probabilistic Cell Decompostition",Proceedings of the IEEE Inetmational Conefernce on Robotics and Automation(ICRA), New Orlenas LA, 2004.

[15] Morten Strnadberg."Robot Path Plnnaing:An Object-Oriented Approach",Automatic Control Department of Singals, Sensors and Systems Royal Institute of Technology, 2004.