Introduction into maze mapping and the shortest path finding

[1] *Ján JADLOVSKÝ*, [2] *Michal KOPČÍK*, [3] *Simona SEGIŇÁKOVÁ*

Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovak Republic

[1]jan.jadlovsky@tuke.sk, [2]michal.kopcik@tuke.sk,
[3]simona.seginakova@student.tuke.sk,

Abstract— **Maze solving has gained increasing attention in the field of Micromouse competition and intelligent robotics. This paper describes some of the methods for maze solving and shortest path finding. This article also contains description of mazes used in mobile robot competition called Micromouse competition. To test proposed algorithms there were created application that simulates mobile robot motion trough maze and also simulates sensor vision which detects the walls of the maze.**

***Keyword* — maze solving, micromouse competition, mobile robot, simulation**

I. INTRODUCTION

Maze solving was always fascinated not only in real world, but also in informatics and robotics. Every year there are organized many robotics competitions where robots are put into maze to solve it in the shortest time. These competitions are called micromouse competitions and have strict rules on the maze and robots. In Micromouse competition contestants compete with their own built robots which they program to move through maze, map it and find the shortest path from start to finish.

The first part of this paper describes Micromouse competition, the maze and robots that can compete. Second part is devoted to basic maze solving methods such as Wall following or Flood fill. The final part deals with developed simulation and visualization tool for testing maze solving methods.

II. MICROMOUSE COMPETITION

Every year, in the spring months, the Faculty of Electrical Engineering and Information Technology in Bratislava organize robotics event called ISTROBOT. In this competition there are several disciplines, in which contestants compete with their robots such as Linefollower, Micromouse, Ketchup storekeeper and Free run. In the Micromouse discipline the challenge is to design and construct an autonomous mobile robot controlled by microcontroller, which can travel through the maze in the shortest time.

A. Rules and Restriction on Maze

The maze consists of a network of squares (8x8 or 16x16 cells), with dimensions of 18x18 cm. The walls of the maze are 5 cm tall and 1.2 cm thick (+/- 5%), so the corridors are 16.8 cm wide. The whole maze is closed by outer wall [1].

The starting position of the robot is in one of the four corners. The destination (goal) is situated in the center of the maze and consists of four unit cells, among which there are no walls. In the maze there are usually several ways from start position to destination, but the entrance to this square is just one. The corners of the individual cells form a points of square grid [1].

A. Rules and Restriction on Robot

The length and width of robot can´t be more than 25 cm. If the mouse is changing its dimensions during operation, at any time the robot may not exceed 25 × 25 cm. The height pf the robot is not limited. While moving through the maze, mouse cannot leave or lose anything behind. The mouse can´t jump across or climb the walls, draw on the walls or floor or damage the maze [1].

During the competition, the time that the robot can spent in the maze is limited to five minutes and has total 10 attempts from start to finish to achieve the best time. Run from start to finish is called as competing attempt and the robot with the shortest time wins the competition. The competition attempt begins to measure by robot leaving the starting cell and ends with his entrance to the destination. After the completion of such a route, the robot can independently go back to the start and try to find shorter path [3].

Some of the rules of this competition are implemented designed application which simulates motion of the mobile robot through the maze, for example dimensions of the maze (8x8 or 16x16) or the position and restrictions on start and finish.

There are several mobile robots and robotics kits, which are commercially available and can be used in Micromouse competition such as:

- Airat 2.
- Robo-Lefter.
- Khepera III.
- LEGO Mindstorms.

III. Maze Solving Methods

In this chapter there are described some of the ways how to solve mazes, which were designed and tested using created simulation application. There are three methods for maze solving described in this article namely:

- Wall follower method.
- Random mouse method.
- Flood fill method.

A. Wall Follower Method

This method of maze solving can be done using two rules, namely Right hand rule and Left hand rule. These two rules are almost the same. The basic principle of this method is monitoring the walls on your right (left) side. The mouse in the maze always chooses the path at the intersection which is the most right (left) one. There are mazes that are not possible to solve with Wall follower method, causing mouse to loop forever along the same path. Example of such a maze is shown in Fig. 1 [2].

Fig. 1 Maze solving using basic right hand method.

To avoid infinite loop that may occur, the improvement of basic Wall follower method is needed. This improvement is based on marking all visited cells by the value that represents how many times the robotic mouse visited that cell. At the start, each cell has zero value and the decision where the robot will go primary depends on the value of the surrounding cells and secondary on the position of the cells around intersection. It continues in this way until they reach the goal. Fig. 2 shows solving maze using improved Right hand rule, where the basic Right (Left) would loop forever.

Fig. 2 Maze solving using improved right hand method.

B. *Random Mouse Method*

This method simulates browsing the maze, as the real mouse will go through it. At each intersection the mobile robot choose randomly from available paths, if there is nowhere to go, robot will turn back and return to nearest intersection. It is possible, that the mouse will visit the same places multiple times, but eventually the mouse will solve the maze.

Reach the goal, can sometimes take a short time, but sometimes it can take very long time, therefore, this method is not suitable for solving larger mazes. This solution however, is a fundamental means of maze solving, so it was designed for demonstration in application. Fig. 3 shows two trials of passing the maze using Random mouse method and it is a nice illustration that the mouse can get to the finish sometimes even faster than using others, more complex methods, but sometimes it can last a very long time. This method however can be improved in the same way the Wall following method was improved in previous chapter by rating individual cells by number that represents how many times the cell was visited. Mobile robot then at the intersection chooses the cell with lowest values and if there are multiple cells with the same value, he will choose randomly one of them. [2].

Fig. 3 Two runs of maze solving using Random mouse method.

C. *Flood Fill Method*

This method is based on calculating of the distance from the end to actual position. This calculation of distance takes in count only the discovered walls, and after each step the robot improves the map of known maze. The walls from unvisited cells are considered as if they were not there. The calculated distance is represented by the minimal number of steps from actual cell to the goal. After calculating the distance at each step the mobile robot chooses the cell with the lowest value.

This method can be used only in the case when the position of the goal is known. Flood fill won't guarantee to find the shortest path at the first time. To ensure to find the shortest path, mobile robot need go through maze multiple times, so he can discover possible shorter paths which weren't found at the first attempt. Solving maze using Flood fill method is illustrated in Fig. 4, more information about this method can be found [3].

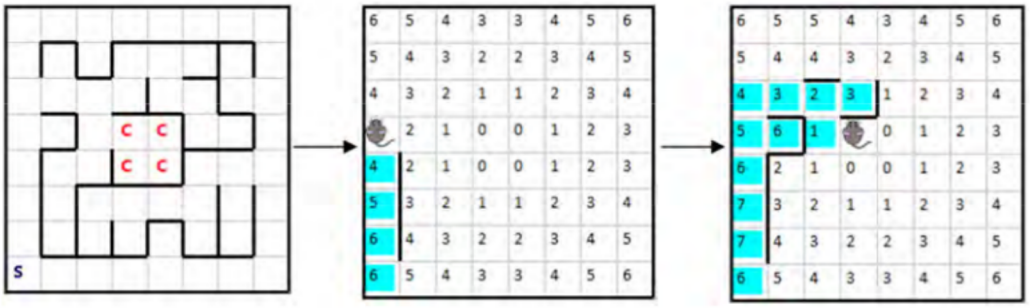

Fig. 4 Example of solving maze by using Flood Fill method.

IV. Simulation Tool for Maze Solving

To test proposed methods for maze solving, there was created simulation and visualization application. The application was developed in object programming language C# in Visual Studio 2013 and consists of two main modules:

- Module for creating the maze.
- Simulation and visualization module.

This application allows simulating mazes with dimensions of 8x8 cells or 16x16 cells, which have start in the corner and goal in the center of the maze.

A. *Module For Creating the Maze*

To create mazes which are later used for simulation, there was developed module using which user can insert or remove walls. The module can create both 8x8 cell and 16x16 cell

mazes. The module allows loading and saving mazes in the form of text file. The form of coding the maze in the file was designed in the form which ensures the user can read the file even without the application. Example of maze and its representation in text file is shown in Fig. 5. Individual walls and nodes are represented by standard console characters.

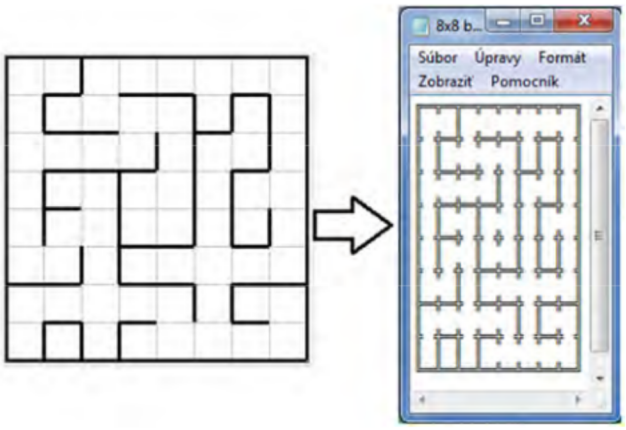

Fig. 5 Sample maze and its representation in text file.

B. Simulation and Visualization Module

On the left side there is loaded maze and on the right side is the maze discovered by the robotic mouse. This simulation application allows controlling the motion of the robot and detection of the walls using simple commands (functions). On the most right side there are buttons which serves for choosing solving method, for selection of speed of the robot and table which holds information about distance travelled from start to finish. On the top side there are buttons to load the maze, draw or edit the maze and button to restart the run of the robotic mouse. The screenshot of designed application is in Fig. 6.

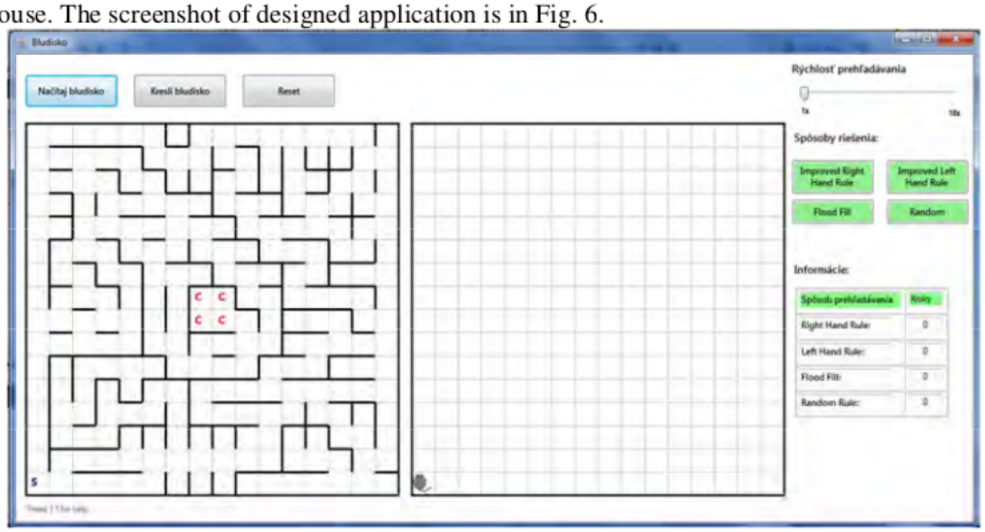

Fig. 6 Main window of created simulation application.

In this application, the user can load the maze from text file. The loaded maze will be plotted in graphical interface. Further, the user can choose one of the options of maze solving:

- Improved Right hand method.
- Improved Left hand rule method.
- Random mouse method.
- Flood fill method.

The user can monitor the progress of these methods in graphical window and also in table which shows actual number of steps done from start. When the user loads another maze or clicks reset button, the robotic mouse will automatically return to starting position and facing up.

V. Conclusion

Experimental verification of the proposed methods was made on six mazes, three of them had dimensions of 8x8 cells and three of them had dimensions of 16x16 cells. Each of these mazes had different characteristics and was used in the Micromouse competition in the past.