2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010)

An Algorithm of Micromouse Maze Solving

Jianping Cai, Xuting Wan, Meimei Huo*, Jianzhong Wu
School of Computer Science
Zhejiang University City College
Hangzhou, China
e-mail: caijp@zucc.edu.cn

***Abstract*—This paper proposes a maze exploring algorithm named "Partition-central Algorithm", which is used to find the shortest path in a micromouse competition maze. A standard 16*16 units maze is divided into 12 partitions in this algorithm. Depending on the absolute direction of the micromouse and the locations of each partition, exploring rules alter when the micromouse walks to optimize the maze exploring process. A simulation program is developed to verify the algorithm. The test result shows that Partition-central algorithm has higher average efficiency when compared with other algorithms.**

Keywords- Partition-central Algorithm; Micromouse; Maze; Simulator

I. INTRODUCTION

The micromouse is a wheeled robot controlled to "walk" inside a maze which uses an embedded micro-control-chip as its brain with artificial intelligence ability. It is equipped with sensors and electrical and mechanical moving parts as well. The micromouse must know its location (coordinate) inside a maze by itself at any time, perceive the walls state around and assign to the defined variables, and eventually complete the competition in the shortest time by processing the stored data and selecting the best path intelligently.

A maze is composed of 256 (16 row * 16 columns) square units, each unit 18cm*18cm in size, The starting point of the maze is on one of the four corners, and the end point is located in the 4 units in the center of the maze and leave only one entrance. The requirements of maze walls and support platform are provided in the IEEE standard [1].

A micromouse has the basic action to walk from the start point to the end point in a competition, which is called one "Run" process. The time is known as the "Run-time", which excludes the time back from the end point to the starting point. The time from the activation of micromouse to the first running process is called "Maze-time". If the micromouse in the game needs to be manually-assisted, this action is known as "Touch". The competition score is calculated with these three parameters, namely, run time, maze time and touch counts. The micromouse capability is evaluated from the speed of solving the maze, the efficiency and reliability of the micromouse [2-4]. Another important terminology is the 'sprint time', which is the time that the micromouse run to the target point with its full speed.

The score in micromouse competition is calculated as the following:

Total time= maze time * factor + sprint time + touch caused penalty [5].

A micromouse must explore the maze to find the path leading to the end point, and keep its location information by itself, sense walls status and calculate to find out the best path depending on the information in its memory, solve the maze and eventually walk along the shortest path to win the competition. In order to find out the shortest path, a certain optimization algorithm must be set for the micromouse to search the maze and find the end point, then reach the terminal in the shortest time. The efficiency of the algorithm determines the efficiency of solving the maze [6]. This paper presents an optimization algorithm which divides a standard maze into multiple partitions, and in each partition different search strategy is adopted according to the current absolute direction of the micromouse.

In the following section, some classic algorithms and the newly designed algorithm Partition-center algorithm are both described. In part 3, a simulator used for verifying and examining the efficiency of micromouse maze searching algorithm is introduced in part 4.

Then, experiments are made to verify and compare the efficiency of 6 kinds of algorithms in part 5. Finally, we evaluate the algorithm by comparing it with other algorithms and give a conclusion.

II. ALGORITHM DESIGN

A. The description of General Micromouse Maze algorithm

In a micromouse maze competition, the competition score results from the maze searching time (maze time) plus sprint time according to the competition rules. A micromouse can only partly explore the maze when traversing it in order to get a good competition result. That means only limited time detection or partial maze detections are available in order to explore the shortest path. To walk in the track in the maze, the micromouse has only a maximum of 3 directions to choose, i.e. the front, left and right, If a micromouse encounters multiple directions crossway, the following rules are available for choosing:

- Left-hand rule: take precedence to the left direction, then straight direction, finally turn right direction.
- Right-hand rule: take precedence to the right direction, then straight direction, finally turn left direction.

* corresponding author: huomm@zucc.edu.cn

978-0-7695-4108-2/10 $26.00 © 2010 IEEE
DOI 10.1109/CIT.2010.337

- Central-left rule: take precedence to straight, then turn left, finally turn right.
- Central-right rule: take precedence to straight, then turn right, finally turn left.
- Central-trend rule: the micromouse always takes precedence to the end point which is located in the 4-unit square area on the maze center [7].

Regular algorithm (rule) strategies behave relatively Non-optimal when exploring a maze. Sometimes the end point is not able to be found during the searching process if such a simple algorithm is executed which leads to a failure in the competition. It is important to shorten searching process by refining and optimizing the algorithm and to further improve the micromouse intelligence level.

B. Maze Information Recording

To describe the algorithm, basically used variables are listed below:

The micromouse absolute direction marking (Dir): a 4-bit binary number is used to represent the micromouse absolute direction which is unrelated with the micromouse current moving direction. if Dir is set to be one of the values of 0x01, 0x02, 0x04, 0x08, the micromouse is to take the next direction of east, west, south and north direction.

Wall data recording (MapBlock [x] [y]): A 16*16 byte-array is defined to store information on the maze walls in the micromouse program.

When the micromouse reaches a grid, current information of the cell walls around is recorded according to sense states of the IR sensors. Each bit is used to represent one side wall status. Value 0 means there is a wall on this side while 1 means not. Each LSB(bit3 ~ bit0) of the byte variable is used to store information of the walls in 4 directions, that is east, west, south and north wall separately. All the variable bits are initialized to 0, indicating there are walls around all units on the maze.

Virtual Wall Recording (VW): The wall state is set to 0 when the micromouse encounters a dead end or a closed looping path during the exploring process.

Times of Micromouse passes a unit (counts[x][y]) : All the elements value is initialize to 0. Each time when the micromouse pass certain unit, count value of this unit increases 1.

C. The contour map Generation

After the micromouse explores in the maze for limited times, a 2 dimension maze chart describing the maze state is formed in the memory of micromouse by marking and calculating the stored information. There may be several paths available from the start point to the end point among the stored information. The micromouse is able to find the shortest path by drawing the contour map from the 2 dimension maze chart [8]. As is shown in Figure 1, the contour map is intended to tell the micromouse the distance step counts of each unit from the state point.

In the contour map, the step counts of each unit to the starting point are known along the feasible path. The path that micromouse takes to return from the end point to the start point is able to be generated automatically, that is, according to the Descending Principles of the unit value on the contour map.

From the starting point on the available forward path, the micromouse takes attendance of walking along an Incremental contour value direction until the end point. Choosing the least step counts to sprinting is one of the important considerations for a micromouse to finish the whole-trip walk and achieve a better score. Meanwhile, it is also critical to consider the less turn path to save time of changing directions, and it should take priority when the whole walking path finally determined.

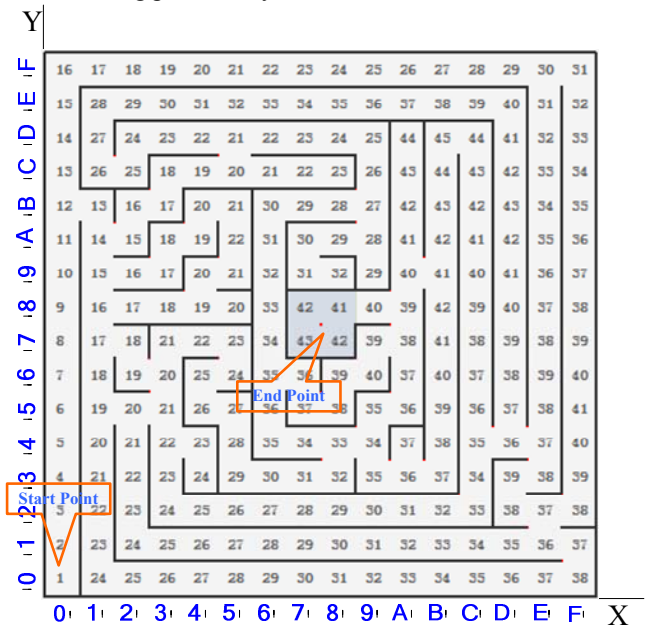

Figure 1. Maze contour map and coordinates.

D. Partition-central Algorithm

By anglicizing the maze search algorithm, based on the conventional algorithms, a new optimized maze exploring algorithm named “Partition-central algorithm” is proposed.

The thought of this algorithm is shown in figure 2. The maze is divided into 12 partitions besides the 4–unit square end point, the 12 partitions division is an optimized scheme after many experiments. The closest to the maze center direction is to be chosen for a micromouse to walk in each partition. Within a certain partition, a rule similar to Potential Value Algorithm is to be adopted [9], i.e. in different partitions, according to the absolute direction value of ahead, the micromouse is set to choose the rule as far as possible toward the direction closest to the end point. The available algorithms are Left-hand algorithm, Right-hand algorithm, Central-left algorithm and Central-right algorithm. By setting the Partition-central algorithm, one of the four algorithms is chosen for a micromouse to explore the shortest path form the start point to the end in each step.

When the micromouse is located in one of the 4 partitions from partition 1 to partition 4, the algorithm selection rule is similar to the Center-trend rule. For example, when the micromouse is in partition 1, and its absolute direction is toward the north, center-right rule is

selected to be the rules of the closest one to the end point; while the micromouse has an absolute direction of toward the east, center-left rule is the best one to the end point; similarly, while the micromouse's absolute direction is toward the south, left-hand rule is the best one to the end point; while the micromouse's absolute direction of toward the west, right-hand rule is the best one; When the micromouse is located in one of the partitions from partition 5 to partition 12, the algorithm determination is quite different from central-trend rule. For example, when the micromouse is in partition 12, if its absolute direction is toward the north, right-hand rule is the rules of the closest one to the end point, (if central algorithm is used, central-right would be selected similar to partition 1 situation); while the micromouse has an absolute direction of toward the east, center-left rule is the best one to the end point; similarly, while the micromouse's absolute direction is toward the south, left-hand rule is the best one to the end point; while the micromouse's absolute direction is toward the west, right-hand rule is the best one. Walking rules of other partitions are shown in Figure 2. From the description above the Figure 2, the Partition-central algorithm is more central-trend than other algorithms.

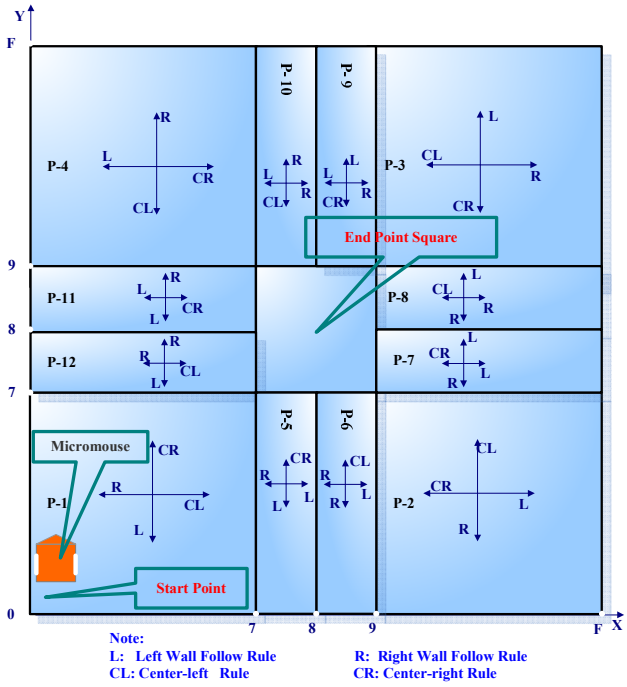

Figure 2. Maze partitions division.

In figure 2, the arrows points to the possible direction when the micromouse is in this partition unit. The letter represents the available rule for choosing.

E. Closed Looping Path Avoidance

The optimized partition-central algorithm is more flexible than other classic algorithms. However, because of the special characteristic of central–trend, after the partition division of the maze, if there is a "T" crossing or a "+" crossroad just between two or among three different adjacent partitions is encountered, different walk determine rules in different partitions would confuse the micromouse and then cause a process of exploring the partition border area repeatedly, thus the micromouse would be led to a searching closed looping path and making it impossible to reach the end point. For example, in Figure 3 (a), without closed looping path avoidance treatment, when entering from A to B, the micromouse prefers the left hand rule, then C, from G back to B over D-E-F. The micromouse is now walking from partition 4 to partition 10 and has a direction towards the east. Following the thought of original Partition-central algorithm and the walking direction, the micromouse will choose the right-hand rule and go back to C, thus causing a repeated walking of closed looping path.

Figure 3 (b) shows another closed looping path situation of maze exploring. It is clear that the portion is located on the adjunction of partition 4 and partition 10 from the coordinates. The unit A is a "+" crossroad unit, so the micromouse has 3 direction-choice when moving from B to A. Meanwhile, the current unit is located in partition 4 and current walking direction is south. If C, D and E have the same count value, central-left rule is adopted, and the micromouse makes C as its preferred direction and then moves F-E-A to D. Next time, if the micromouse moves again from B to A following the partition-central algorithm, closed looping path would emerge.

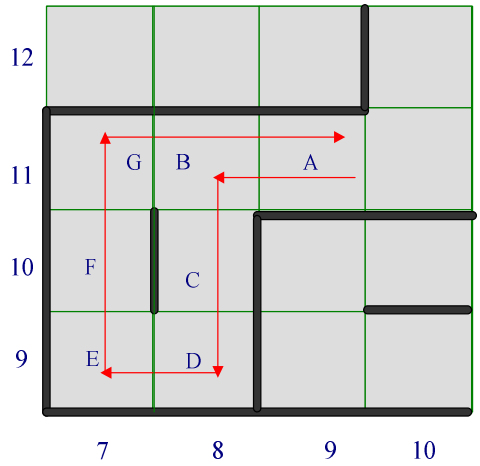

Figure 3 (a). The first type closed looping path.

The following section shows how to solve the closed looping path problem by using the two variables: virtual wall and step counts which are introduced in section 2.2.

In the "T" crossing situation, the problem is solved by using the two variables in the program. From Figure 3, the maze portion is located on the adjunction of partition 3, partition 9 and partition 10. The micromouse enters partition 9 after moving from A to B, and the count of B is assigned to

count+1. The current walk direction is the east so the left-hand rule is set. Then the micromouse arrives G over D. When it enters B from G, if the count value of B is not 0, the program would check if the unit next to B has been visited or not before. In Figure 3(a), the program will check the count of C. If the value is 0, the micromouse will pass B to C, if not, another direction (not B then C) is to be selected. In this case, the count value of C is clearly not 0, so the micromouse selects the path of from B to A rather than from B to C, at the same time it sets virtual wall (VW) on B to avoid the closed looping path again.

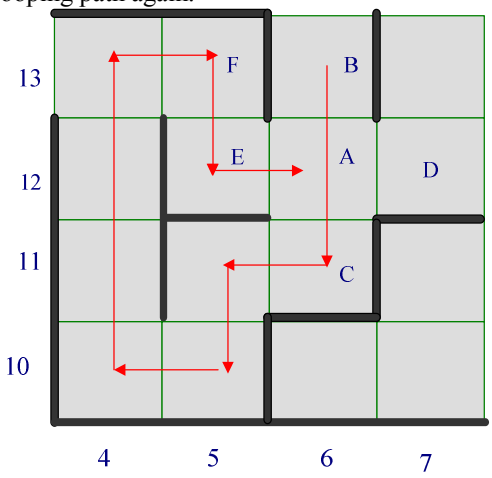

Figure 3 (b). The second type of closed looping path.

F. Flow of maze solving with the Partition-central algorithm

The flow chart shown in Figure 4, the steps are described as the following:

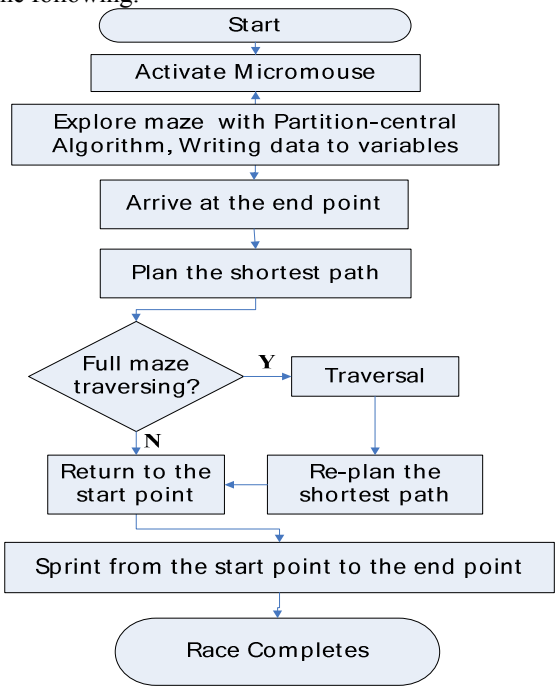

Figure 4. Flow chart of a micromouse walking.

- Exploring the maze from the start point with the partition-central algorithm until the end point to record the path information and wall information of the maze.
- Generating the contour map of the maze, and calculating the shortest path to the end point. In this step, Flood fill algorithm can be used to determine the shortest path.
- Arriving at the end point by applying the partition-central algorithm, then returning to the start point updated with Flood fill algorithm, or searching the full maze until it finds a better or shorter path.

Making a sprint run from the start point, and following the best path found at full speed until the end point. After a short pause according to the competition rules the micromouse returns to the start point to complete the competition race.

In a micromouse competition, the micromouse is asked to hand in to the committee before the maze map is announced. The operator is not allowed to make any change to the competition micromouse since then, including replacing the hardware, downloading software or altering algorithms. During the running process, any failure may cause a re-walk from the start point, and the total time cost will be added together as "maze time" for penalty. In 2009 Beijing Micromouse Competition, the rule is 10 seconds award will be canceled for the first touch which is caused by running failure, and the second touch will be added 5 more seconds for penalty. So a careful calculation to the direction turning is also important besides an excellent algorithm.

The total time that a micromouse is allowed to stay in the maze is 15 minutes. Running for any times is acceptable within the time, but it is meaningless because the score maybe very low.

III. SIMULATION SOFTWARE

In order to verify and test the feasibility and reliability of self-designed algorithms, a simulator is developed to simulate the micromouse maze searching and running process. The program is developed with Visual C + +. General function is provided both in the menu list and the buttons. Maze data files are designed in plain text format and easy to store and load. The following features are realized currently:

- Freely manual constructing a 16 × 16 standard micromouse maze;
- 6 maze exploring algorithms being embedded inside currently for choosing to explore the maze and record the maze information;
- Generating the contour map to any standard maze.
- Displaying the Searching steps and sprint steps real timely;
- Saving and loading maze map file in plain text file;
- Displaying contour map up to the maze map.
- Adjusting animation speed when a micromouse moves in the maze.

In the simulation program, interfaces are left for new algorithm to be added, any newly developed algorithm can be directly connected to the simulator.

The simulator brings lots of convenience when the micromouse is debugged. The optimized algorithm can be added to the micromouse program easily as well.

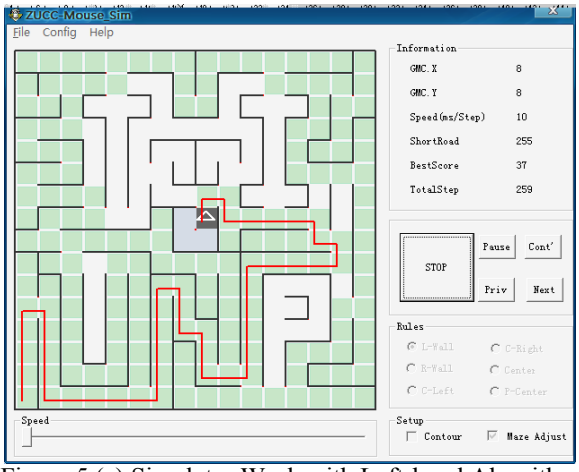

Figure 5 (a) Simulator Work with Left-hand Algorithm

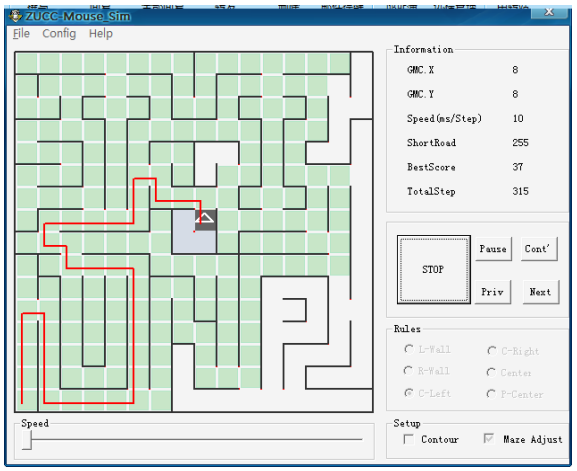

Figure 5 (b) Simulator Work with Center-left Algorithm

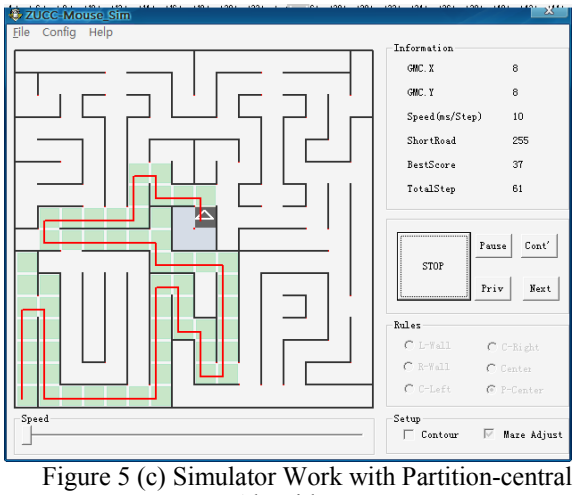

Figure 5 (c) Simulator Work with Partition-central Algorithm

The Figure 5(a), (b) and (c) show, the simulator is also fit for algorithms comparison. With the same TI Unpar maze, the micromouse with the left-hand algorithm needs 259 exploring steps, and the center-left algorithm needs 315 steps while the Partition-central algorithm needs 61 steps to find out the same 37 steps shortest path.

IV. ALGORITHM TESTING AND VERIFICATION

In order to test and validate the advantages of the self designed partition-central algorithms, 7 important competition maze maps are selected for comparison. Table 1 lists the UK, LW, ST, TI, KATO and Beijing competition maze maps to verify the maze solving algorithm [1-5] [8][9][10][11][12].

TABLE I. A comparison of the Exploring Steps.

Rules	Steps	UK	LW	ST	TI UNP	KATO	BJ
Partition central	**Exploring**	**169**	**135**	**91**	**61**	**171**	**113**
Center-left	Exploring	367	277	351	315	181	205
Central	Exploring	141	299	109	61	293	113
Left-hand	Exploring	261	119	311	259	147	135

TABLE II. A comparison of the Shortest Path Steps.

Rules	Steps	UK	LW	ST	TI UNP	KATO	BJ
Partition central	**Shortest Path**	**41**	**71**	**47**	**37**	**63**	**73**
Center-left	Shortest	41	71	47	37	63	73
Central	Shortest	41	71	47	37	63	73
Left-hand	Shortest Path	41	79	47	37	73	135

Table I and Table II show the micromouse efficiency performs distinctly while different exploring algorithm in the same algorithm is loaded. There is also big difference while the same algorithm is used to explore different mazes. The difference of the efficiency is very obvious. However, the result also shows that partition-central algorithm has a relatively high efficiency in exploring the maze and finding the shortest path in most of the mazes.

V. CONCLUSION

Performance efficiency varies in a big range with different exploring algorithms applied to different maze maps. An optimal search algorithm can hardly be found universally applicable to fit all kinds of maze to get maximum efficiency. But the exploring time cost can be shorten to search the best path to win the match with special optimized algorithm.

The partition-central algorithm divides a maze into 12 partitions, and applies different rules to different small areas; the exploring process is more flexible and improves the

intelligence of a micromouse. The partition-central algorithm performs more central trending feature than classic central algorithm, and its advantage shortens the maze exploring time to some extent, which is verified in most experiments that have been finished; meanwhile, shorter exploring time reduces possible collision to the maze wall when the micromouse running. In turn the micromouse is able to traverse the maze to reach destination faster.

A 2-dimensional micromouse simulation software is developed for micromouse debugging and competition preparation purpose, which is helpful to optimize algorithm in practice.

Users can easily finish algorithm verification and algorithms cooperation comparison work on a computer rather than in the real maze, thus lots of test time can be saved. Different to standard simulation software products, the self-developed simulator has lost of advantages such as adding newly developed algorithms to the simulator to verify and compare the efficiency with all existing algorithms.

Micromouse competition is contest of comprehensive ability to participants. In addition to artificial intelligence algorithms and other programming abilities, participants also are required to have wide range of experiences and practical abilities such as embedded system applications, sensors applications, automatic control technology and artificial intelligence technology. But once a good micromouse is designed and manufactured, the algorithm is the critical factor to the competition performance.

Micromouse Maze competition is an events related to testing, artificial intelligence, automation, computers machinery and a lot of other knowledge. It is helpful for students' creativity training and motivating the reform of computer science and information electric education.

REFERENCES

[1] http://Ewh.ieee.org/reg/1/sac/Main/.../MicromouseRules.pdf

[2] http://micromouse.cannock.ac.uk/rules.htm

[3] micromouseInfo.com, http://www.micromouseinfo.com/

[4] micromouse UK (2004), http://micromouse.cs.rhul.ac.uk/

[5] http://www.micromouse.com.cn/dasai/jieshao.asp

[6] Manoj Sharma, Kaizen Robeonics, Algorithms for Micro-mouse, 2009 International Conference on Future Computer and Communication

[7] Zhang xinyi, A Micromouse Maze Solving Algorithm. MCU and Embeded System, 2007.5: 84-85.

[8] University of California, Berkeley, IEEE Student Branch (2006), http://ucsee.eecs.berkeley.edu/

[9] Bagus Arthaya, Ali Sadiyoko & Ardelia Hadiwidjaja, The design of a maze solving system for a micromouse by using a potential value algorithm, World Transactions on Engineering and Technology Education © 2006 UICEE Vol.5, No.3, 2006

[10] Swati Mishra, Pankaj Bande, Maze Solving Algorithms for Micro Mouse, 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems

[11] Babak Hosseini Kazerouni, Mona Behnam Moradi and Pooya Hosseini Kazerouni;"Variable Priorities in Maze-Sloving Algorithms for Robot's Movement", 2003.

[12] David M. Willardson. Analysis of Micromouse Maze Solving Algorithm [R]. Learning from Data, spring 2001