A Simple and Efficient Diagonal Maze-solver for Micromouse Contests and intelligent mobile robot education

Juing-Huei Su[1], Hsin-Hsiung Huang[1], Chyi-Shyong Lee[1]

1. Department of Electronic Engineering, Lunghwa University of Science and Technology, Taoyuan, 33306, Taiwan
E-mail: suhu@mail.lhu.edu.tw

Abstract: A simple and efficient time-based diagonal maze solver for classic and half-size micromouse contesets is presented in this paper. The algorithm is first developed in the graphical user interface development environment (GUIDE) of MATLAB, and then used as a training and tutorial tool for students who are involved in contests oriented projects about intelligent mobile robots. It is also found that the firmware implementation of the algorithm in a dsPIC 16-bit microcontroller consumes just 4.2ms to finish searching for the best route from the start cell to the goal area in a 16x16 maze of classic micromouse contests.

Key Words: Diagonal maze solver, Micromouse, Intelligent mobile robots

1 INTRODUCTION

Robots are playing an increasingly important role in exploration and our daily life, especially when the exploration rover, Curiosity, landed on Mars [1], and iRobot introduced the vacuum-cleaning robot Roomba [2]. Understanding the construction and control of mobile robots has therefore become a necessity for many electrical and electronic engineers. Unforunately, the robot design draws on many areas of knowledge, such as mechanics and electronics, automatic control theory, and software programming of microcontrollers [3-4]. Nevertheless, students are willing to do tedious research work to solve practical problems when these are related to an interesting, competitive contest [5-7]. Winning one or two awards in a competition not only gives students a sense of accomplishment, but also gives their schools pride and visibility. This is an important factor for technology-oriented university students in Taiwan, who tend to have low learning achievements in traditional theory-oriented lecture courses. The original Micromouse contest, in which autonomous robots compete in terms of speed and intelligence, were started about 30 years ago. Because the robots' performance is still improving, these contests are still very popular with engineering students in the U.K., US, Singapore, Japan (Fig. 1) [8], and Taiwan [9]. To challenge the contestants even more, the rule was changed in Japan in 2010 to adopt a larger maze (32×32) and smaller maze cell dimensions shown in Fig. 1. Micromouse is basically an autonomous mobile robot that has to search from the start cell to the goal cell in an unknown maze for the shortest path. The goal cell is fixed in the center for classic contests (16×16), and can be any area in the maze for the new half-size contests. The micromouse has to run as fast as possible from the start cell to the goal area to win the contest. Although the objective seems simple enough, the contestants have to devise an efficient maze-solving algorithm such that the goal area can be found within a time limit. Because of the fast development of microcontrollers and MEMS sensors, the micromouse can be built to be small enough that it can run diagonally in the maze. This poses a new problem to traditional maze solvers whether or not these solvers can find the best diagonal route from the start cell to the goal area, and take into account the motion capabilities of the micromouse.

Fig 1. The 32nd All Japan half-size micromouse contest held in Tsukuba.

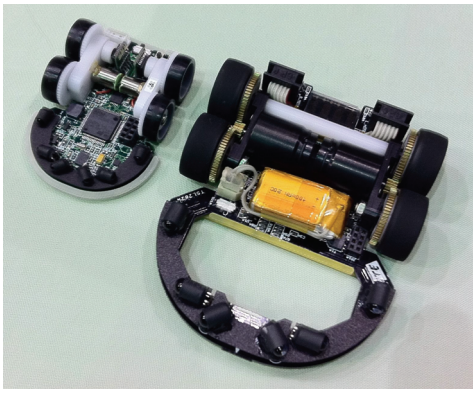

Fig 2. Micromouse for classic (right) and half-size (left) micromouse contests made by Khiew, Tzong-Yong from Singapore

This work is supported by National Science Council of Taiwan under Grant NSC 102-2221-E-262-016.

Therefore, it is the goal of this paper to devise a simple and efficient diagonal maze solver which takes into account the motion capabilities of micromouse robots, and to use the algorithm to motivate students to explore the knowledge about intelligent mobile robots.

2 A TIME-BASED DIAGONAL MAZE-SOLVER

In this section, a time-based diagonal maze solving and path planning algorithm and its corresponding behavior model simulation is described.

2.1 Diagonal flooding algorithms

Traditionally, flooding algorithms are very popular in finding the goal and best route of a given maze. The criteria of these solvers are mainly based on the distance of the route from the start cell to the goal area with right angle turns. The numbers filled in the maze shown in Fig. 3 stand for the distance from the cell to the start, and the best route is found from the goal area following a descending order of numbers to the start cell. Although there are 4 paths in Fig. 3 from the start cell to the goal that are suggested by a distance-based flooding algorithm, the numbers of right angle turns for these paths are different.

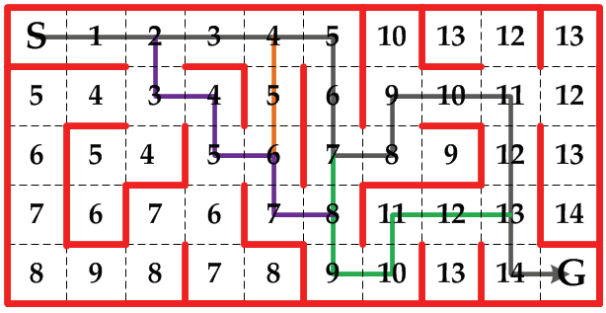

Fig 3. A distance based maze solver by using flooding algorithm [10]

Since the approach shown in Fig. 3 does not take into account the motion capabilities for a micromouse to run in a diagonal path shown in Fig. 4, the flood values in each cell can be moved to borders between cells to remedy this problem. The modification is shown in Fig. 5.

Although the simple distance based diagonal maze solver shown in Fig. 5 can find a unique diagonal path from the start cell to the goal, it still does not take into account the differences of motions for a micromouse to run in a straight path or to make turns. The algorithm may fail to identify a path with more and longer straight paths in a maze from the start cell to the goal area.

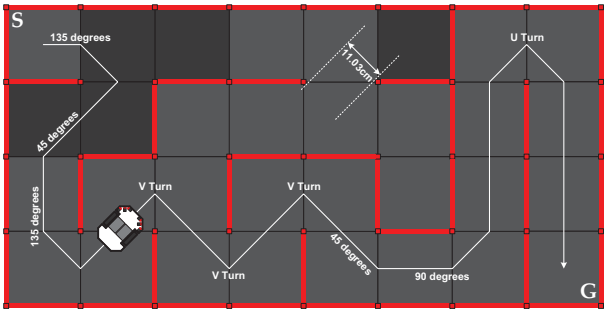

Fig 4. Diagonal motion capabilities of micromice developed in these years

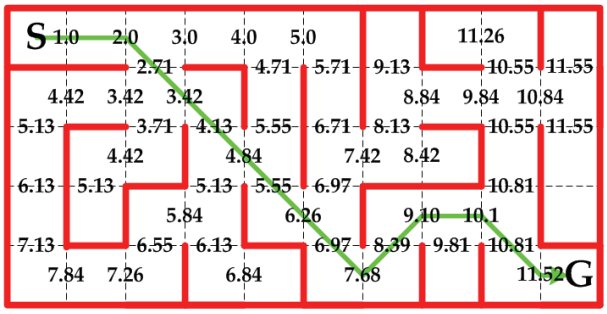

Fig 5. A distance based diagonal maze solver

This results in the development of time-based maze-solving algorithm described as follows.

2.2 A time-based flooding algorithm

This approach originates from the idea of considering the motion capabilities at straight and turning paths. If a micromouse can accelerate at straight paths and decelerate to a suitable speed for making turns, the maze solver should try to find a route with more straight paths. Since the flooding values are filled at the borders of maze cells to find diagonal routes, the time calculation for the flooding values would be based on the distance between these borders. Assume that the acceleration and deceleration of a micromouse are 10m/s2 and −10m/s2, respectively, and the turning speeds for 45, 90, 135, and 180 are all 1m/s. The top speed of a micromouse is basically limited by the motor specifications and how well the micromouse can align itself in normal or diagonal straight paths. It is assumed that the top speeds for a micromouse are 3.44 m/s for normal and 2.94 m/s for diagonal straight paths, after 3 cells of acceleration in this diagonal maze solver.

Since the distance between maze cell borders is 18 cm, the time for a micromouse to run across a cell is 0.18s with the turning speed 1m/s. The necessary time for a micromouse to reach different borders in straight and diagonal straight paths can be calculated by using the following formulas:

$$v_{fn}^2 = v_0^2 + 2(10)n \times 0.18,\ n = 1,\ \cdots,\ 3, \tag{1a}$$

$$v_{fdm}^2 = v_0^2 + 2(10)m \times \frac{0.18}{\sqrt{2}},\ m = 1,\ \cdots,\ 3, \tag{1b}$$

$$t_n = \frac{v_{fn} - v_0}{10},\ \Delta t_n = t_n - t_{n-1}, \tag{1c}$$

$$t_m = \frac{v_{fdm} - v_0}{10},\ \Delta t_m = t_m - t_{m-1}, \tag{1d}$$

where v_{fn}, v_{fdm}, v_o, n, m, t_n, and t_m stand for final velocities in straight and diagonal straight paths, initial velocity, number of normal and diagonal cells for acceleration, and the time needed for acceleration in straight and diagonal straight paths, respectively. Since the lowest speed in this case is the turn speed 1m/s, the initial velocity v0 will be regarded as 1 in (1). Table I shows the time value in each cell for a micromouse to run in normal or diagonal straight paths. The normalized flood values for each cell in Table I are obtained by dividing Δt_n and Δt_m with a base value of 0.18s, which is the time value for a micromouse to run across a cell with a speed of 1m/s.

Table1. Flood Values for Straight Path Acceleration

order of cells, n	1st	2nd	3rd	4th
Δt_n for straight path (v_{ftn} in m/s)	0.114 (2.14)	0.072 (2.86)	0.058 (3.44)	0.052 (3.44)
Δt_m for diagonal straight path (v_{ftdm} in m/s)	0.088 (1.88)	0.059 (2.47)	0.047 (2.94)	0.043 (2.94)
Normalized flood value for Δt_n	0.63	0.4	0.32	0.29
Normalized flood value for Δt_m	0.49	0.33	0.26	0.24

To complete the time based flooding algorithm, the penalty flood values when making turns after acceleration for a number of cells should also be calculated. This is based on the observation that the speed for a micromouse should be decelerated to 1 m/s for making turns. Therefore, the following equations are used to find the penalty flood values

$$v_{ft}^2 = v_0^2 + 2(10)\frac{n}{2}\times 0.18,\ n=1,\ \cdots,\ 6, \tag{2a}$$

$$v_{ftd}^2 = v_0^2 + 2(10)\frac{m}{2}\times\frac{0.18}{\sqrt{2}},\ m=1,\ \cdots,\ 6, \tag{2b}$$

$$t_{n,t} = \frac{2(v_{ft}-v_0)}{10},\ \Delta t_{np} = t_{n,t} - t_n, \tag{2c}$$

$$t_{m,t} = \frac{2(v_{ftd}-v_0)}{10},\ \Delta t_{mp} = t_{m,t} - t_m, \tag{2d}$$

where v_{ft}, v_{ftd}, $t_{n,t}$, and $t_{m,t}$ stand for top velocities before deceleration in normal and diagonal straight paths, the necessary time for the micromouse to decelerate to the turn speed of 1m/s, respectively. The following table summarizes the normalized penalty values for a micromouse to decelerate to the base speed of 1m/s after acceleration.

Table2. Penalty values for Micromouse in Making Turns after Acceleration

order of cells, n	1st	2nd	3rd	4th	5th	6th
$t_{n,t}$ (v_{ft} in m/s)	0.135 (1.67)	0.229 (2.14)	0.306 (2.53)	0.373 (2.86)	0.433 (3.16)	0.487 (3.44)
Δt_{np} (normalized)	0.020 (0.11)	0.043 (0.24)	0.063 (0.35)	0.077 (0.43)	0.084 (0.47)	0.086 (0.48)
$t_{m,t}$ (v_{ftd} in m/s)	0.102 (1.51)	0.177 (1.88)	0.239 (2.20)	0.294 (2.47)	0.343 (2.71)	0.388 (2.94)
Δt_{mp} (normalized)	0.013 (0.07)	0.030 (0.17)	0.045 (0.25)	0.056 (0.31)	0.062 (0.35)	0.064 (0.36)

Fig. 6 shows how Table I and II are used to fill flood values at borders of maze cells from the start. The 'S' at the lower left corner stands for the start cell. Since the micromouse first moves from center of the start cell at north direction, the initial flood value at the border of the start cell without maze wall is 0.5. The process is stopped after one of the borders in the goal area is filled with non-zero flood values (shown in Fig. 7). Three kinds of information are stored along the process of filling the flood values, 1) the flood value, 2) the direction from previous to current border, and 3) the number of cells that are in the same direction.

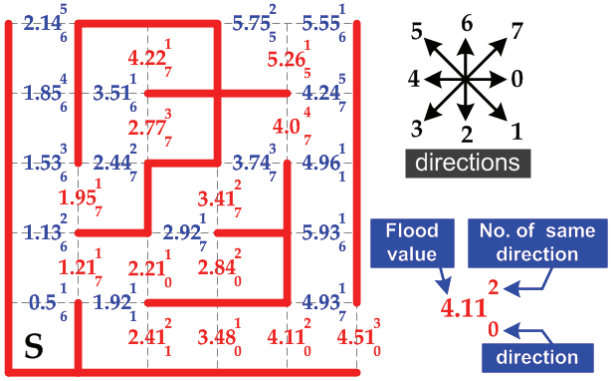

Fig 6. The process of filling flood values with Table I and II

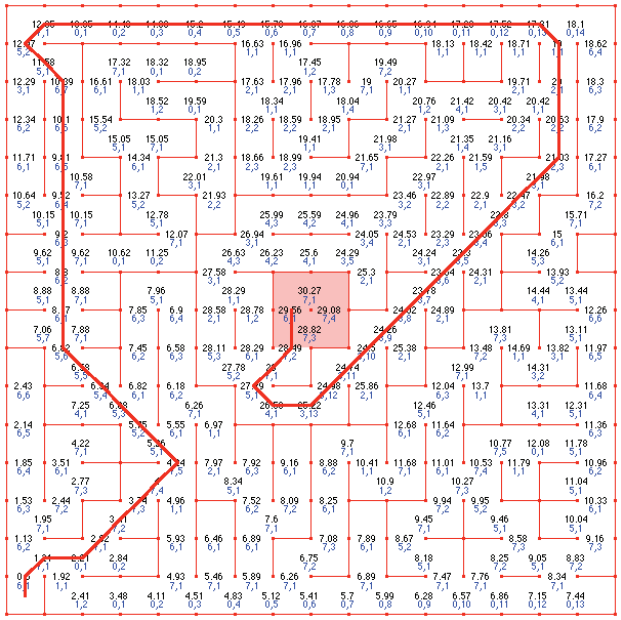

Fig. 7. The flood value information after the goal area is reached

2.3 The optimal diagonal path

Once the goal area is reached in the flood value filling process, the optimal diagonal path can be found according to the flood value information from the start cell to the goal area. Since the flood value filling process originates from the start cell, the optimal diagonal path would be constructed which begins with the largest flood value in the goal area. The next flood value or the border is chosen based on the following criterions:

1. If the number for the same direction k is greater than 1, the next border should be the one with number $k-1$;
2. If the number k at the first step is 1, the next border would be the adjacent one to the current with smallest flood values.

Fig. 7 shows the final result of the devised time-based diagonal maze solver when the 2011 all Japan classic micromouse contest maze is used. It is interesting that the chosen optimal path (red line) seems to be the longest one for distance based maze solvers. All the contestants except BengKiat Ng chose the other paths. This may be the reason why BengKiat Ng broke the 4-second limit (3.921) and won the championship.

2.4 The strategy of searching for the goal

The discussion in the previous sections assumes that the maze wall information in each cell is known for the flooding process. The real situation is that a micromouse has to search for the goal without knowing anything about the maze at first. Therefore, a micromouse would collect the maze wall information of any visited cell along the searching process, and regard those unvisited cells as no wall at all ('open') to get a suggested path to the goal. The strategy of searching for the shortest path from the start cell to the goal area proposed in this paper consists of the following 3 steps:

1. Set the current location of the micromouse as the 'start' position and regard those unvisited cells as 'open' (no wall at all). The diagonal maze solver can then be used to obtain a shortest path based on the 'start' position, the maze wall information, and the goal area location. The micromouse would follow the shortest path to the next cell and collect its wall information if the moving direction of the micromouse at the 'start' position is the sames as that of the shortest path at the same cell (shown in Fig. 8). Otherwise, the micromouse has to adjust its moving direction to follow the suggested shortest path. The micromouse would continue the process until the goal area is found;
2. To guarantee that the true shortest path is found, the shortest paths based on 'closed' (maze wall exists at each border) and 'open' maze wall information of unvisited cells should be the same. The shortest path obtained with 'closed' maze wall information for unvisited maze cells are marked as dashed black line in Fig. 9, and the blue line stands for the shortest path obtained with 'open' maze wall information. Therefore, the micromouse has to check the wall configuration of those unvisited cells that are in the shortest path suggested by the diagonal maze solver with 'open' maze wall information.
3. Return to the start cell at the corner of the maze, when the true shortest path is found.

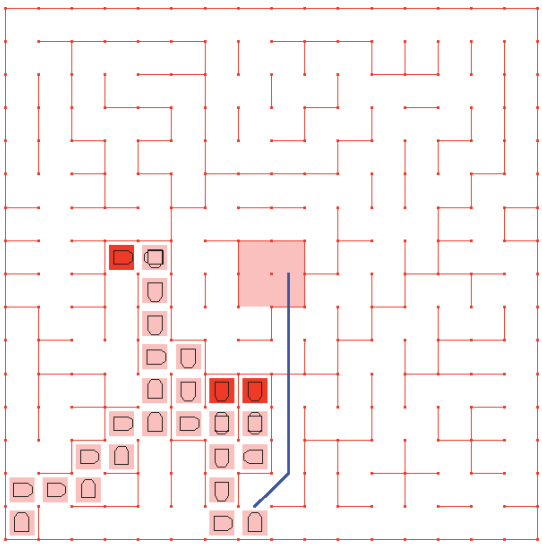

Fig. 8. The searching process of the proposed strategy

2.5 Behavior simulation of the diagonal maze solver

The idea of searching for the goal with the diagonal maze solver is similar to that described in [10], and the entire process can be coded as an m-file in the MATLAB's graphical user interface development environment (GUIDE) shown in Fig. 9. The graphical user interface shown in Fig. 9-10 can not only serve as a visual guide to students, but also give an environment for them to fine tune their searching strategies. It can be seen in Fig. 9 that all visited maze cells are marked as pink, and those red maze cells reveal that the micromouse made 180 degree in-place turns in those cells when searching for the goal. The black dashed line in Fig. 9 shows the shortest path using only those visited cells, and the blue line shows a better possible path if those unvisited cells are regarded as no maze walls around them. Students can devise and verify how well their idea could be in searching the global optimal path in this environment without writing codes directly into the microcontrollers.

The graphical user interface for a more challenging half-size micromouse contest is also developed in Fig. 10. With the help of this environment, students can easily separate the micromouse development into two independent parts, the motion control and maze solver. Therefore, the process of learning can be made more smooth and effective for students who are interested in exploring the knowledge of intelligent mobile robots.

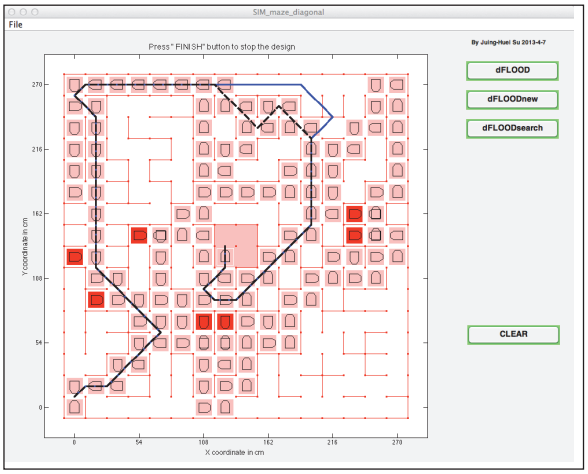

Fig. 9. A graphical user interface for maze solving in a classical micromouse contest

3 FIRMWARE IMPLEMENTATION

The algorithm is first implemented and verified in the GUIDE provided by the MATLAB, and the code is quite similar to that of C-language. The only major difference is that the index number for arrays starts from 1 in GUIDE instead of 0 for C-language. Therefore, it would not be difficult to port the algorithm devised in GUIDE to C-language in any commercially available microcontroller.

The 16-bit dsPIC 33EP512MC806 from the Microchip Inc. is chosen in this paper to find out how fast the algorithm can be executed, because of the free support from the local branch in Taiwan. The second reason for the choice is that the microcontroller will also be used as the main controller for a contest-oriented classic micromouse.

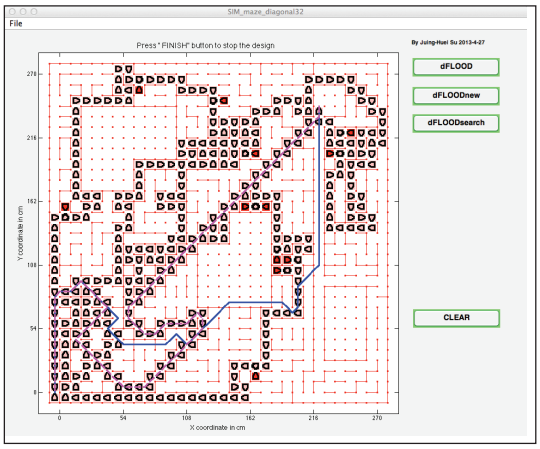

Fig. 10. A graphical half-size micromouse contest diagonal maze solver validation environment

To make the algorithm suitable for the searching strategy described in the previous section and half size micromouse contest, the settings for 'start' cell and goal area should be configured as input parameters for the firmware subroutine. The firmware implementation is accelerated using the idea of 'wavefront' in propagating the flood values step by step to the goal area. Since the 'start' cell is not necessary to be loacated at one of the corners of a given maze when searching for the goal, it is possible that the initial wavfront may consists of more than one borders. This may increase the possibility that the generated shortest path would give a different direction from the moving direction of the micromouse at the 'start' cell. The problem can be remedied by giving penalty values to those initial flood values which are at the borders out of the micromouse's moving direction. The execution time to find the optimal shortest path for the time-based diagonal maze solver, when given the maze of the final contest in All Japan classic micromouse contest in Fig. 7, is about 4.2ms (shown in Fig. 11), and the generated shortest path is the same as the one used by the champion BengKiat Ng [12]. It would take a longer time to finishing executing the algorithm when it tries to use the spare time of an interrupt driven micromouse control firmware. For example, if the interrupt timing is 1ms, and the execution of the control firmware consumes about 70% of the time, the algorithm can only be finished after 4.2/0.3 = 14ms. This corresponds to about 1cm if the searching speed is 70cm/s.

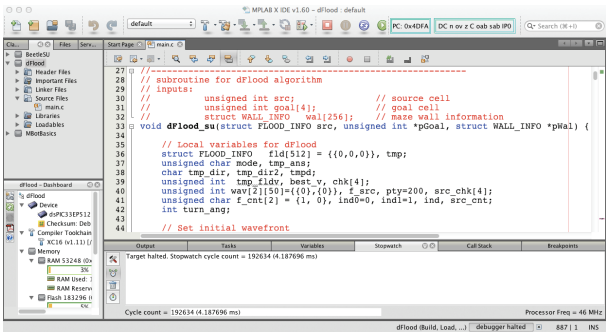

Fig. 11. The firmware implementation of the time-based diagonal maze solver as a subroutine, and its runtime for a given maze.

This would not be a serious constraint for any contestant, because most of the micrommouse robots search the maze at speeds lower than 70cm/s and the distance from knowing the maze wall information of the next cell to the decision point of the next motion pattern can always be made larger than 1cm.

4 CONCLUSIONS

A simple time-based diagonal maze solver for both classic and half size micromouse contests is presented and implemented in this paper. All the complex computations can be done and saved as tables before the algorithm is executed. Therefore, it is efficient enough to be executed within 4.2ms. This corresponds to about 1 cm of distance in maze when a micromouse searches the maze at a speed of 70cm/s, and leaves 30% of time in a 1ms interrupt driven control subroutine to the maze solver. The behavior simulation of a micromouse to search in a maze is also developed as a tutorial tool for those students who are interested in the implementation skills of intelligent mobile robots and micromouse contests. It is also seen that the behavior simulation helps a lot to interest students and motivate them to explore the implementation skills of micromouse contests and intelligent mobile robots.

REFERENCES

[1] NASA's Curiosity Mars Rover, accessed on 7/2/2013, www.facebook.com/MarsCuriosity?hc_location=stream.

[2] iRobot, Bedford, MA, "Vacuum cleaning robot—iRobot Roomba," Accessed Oct. 9, 2011. www.irobot.com.

[3] K. Nagai, "Learning while doing: Practical robotics education," *IEEE Robot. Autom. Mag.*, vol. 8, no. 2, pp. 39–43, Jun. 2001.

[4] K.-S. Hwang, W.-H. Hsiao, G.-T. Shing, and K.-J. Chen, "Rapid prototyping platform for robotics applications," *IEEE Transactions on Education*, vol.54, no. 2, pp. 236–246, May 2011.

[5] N. Chen, H. Chung, and Y. K. Kwon, "Integration of Micromouse project with undergraduate curriculum: A large-scale student participation approach," *IEEE Trans. on Education*, vol. 38, no. 2, pp. 136–144, May 1995.

[6] R. B. Murphy, "Competing for a robotics education," IEEE Robot. Autom. Mag., vol. 8, no. 2, pp. 44–55, Jun. 2001.

[7] D. J. Pack, R. Avanzato, D. J. Ahlgren, and I. M. Verner, "Fire-fighting mobile robotics and interdisciplinary design-comparative perspectives," IEEE Transactions on Education, vol. 47, no. 3, pp. 369–376, Aug. 2004.

[8] New Technology Foundation, Tsukuba, Japan, "The 32nd All Japan Micromouse Contest," Accessed Dec. 24, 2011 [Online]. Available: www.ntf.or.jp/mouse/micromouse2011/index_EN.html.

[9] Lunghwa University of Technology, Taiwan, "The 2013 Taiwan Micromouse and Intelligent Robot Contest," Accessed July 2, 2013. Available: http://robot.lhu.edu.tw.

[10] Society of Robots, "Micromouse – Maze Solver – Theory," Accessed July 12, 2013 [On-line]. Available: website: http://www.societyofrobots.com/member_tutorials/node/94.

[11] H.-H. Huang, J.-H. Su, and C.-S. Lee, "A Contest-Oriented Project for Learning Intelligent Mobile Robots," *IEEE Trans. on Education*, vol. 56, no. 1, pp. 88–97, Feb. 2013.

[12] Youtube, 2011 All Japan micromouse contest: Ng BengKiat 4th Fast RUN, accessed August 6, 2013 [Online]. Available: http://www.youtube.com/watch?v=CLwICJKV4d