World Transactions on Engineering and Technology Education *© 2006 UICEE*
Vol.5, No.3, 2006

The design of a maze solving system for a micromouse by using a potential value algorithm

Bagus Arthaya, Ali Sadiyoko & Ardelia Hadiwidjaja

Parahyangan Catholic University
Bandung, Indonesia

ABSTRACT: Automation has recently influenced the industrial sectors in a wide range. To be able to survive in the competitive world, a firm has to keep increasing its efficiency, effectiveness and productivity. Automated systems have been developed to achieve that objectives and have becoming more applicable in industrial sectors. These systems are designed to ease and make safe difficult and hazardous human tasks such as welding, spray painting, car-body assembling, reactor dismantling and so on. An example of automated systems is Autonomous Mobile Robot (AMR). A computer simulation model of AMR has been developed and a certain degree of intelligence is implanted. This is needed when the robot has to take its own decision in determining the path. The task of this robot is to reach a target region which is located in the centre of a certain labyrinth. The labyrinth has never been mapped before and the robot has to find the easiest way to get to the target. The search should be performed autonomously without any human intervention. Some simulations show the capability of the robot in searching its goal and it behaves differently depending on the complexity of the maze.

INTRODUCTION

Nowadays, automated systems have been intensively explored and implemented to help human beings perform their daily activities. In the industrial field, automated systems support many tasks, such as storing and retrieving goods to and from storage, performing automated welding inspection, automatic painting, etc. People tend to develop manufacturing systems that can be easily adapted to any environment. One application of such a system is normally used for material handling and transportation. An example of this system is an Autonomous Mobile Robot (AMR). The behavioural complexity of this robot is very high. One way to trace it back is by developing a micromouse robot, which is an electro-mechanical miniature of an AMR assigned to perform tasks in an unknown maze. It has a freedom to move around within its environment. The maze consists of a labyrinth and walls that construct the maze all together. The robot is ordered to move from one origin point to the final destination point, which is located in the centre of the labyrinth. In fact, this labyrinth is an unknown maze for the robot that has never visited it before [1-5].

The searching should be performed autonomously without any human intervention, which means an algorithm is needed to enable actions without any human help. Searching algorithms for this kind of task have been explored and improved to reach the target or finish line as fast as possible. This research focuses on developing a searching algorithm based on the implementation of a potential value concept and comparing the results with some existing algorithms used for similar tasks.

PROBLEM IDENTIFICATION

Some problems found here include identifying how to build a searching algorithm in a labyrinth that has never been mapped to reach a goal point and how the design of a simulation system to examine the algorithm looks like.

IEE MICROMOUSE RULES

A micromouse robotic competition has to comply with IEE micromouse standard rules. These competition rules consists of three parts, ie the labyrinth rule, robot rule and competition rule [6][7]. The labyrinth comprises 16 x 16 cells (square form) of 18 cm x 18 cm. The wall thickness and height are 1.2 cm and 5 cm, respectively. The outermost wall covers the whole labyrinth. The coating on the top and wall side are chosen so that it has capability to reflect an infrared beam, while the coating on the floor should absorb the beam.

An example of a valid labyrinth is depicted in Figure 1. The robot starting position is at one cell at the corner of labyrinth (designated as S). The cell must have three walls surrounding it. The starting point (cell) has an opening heading North, with the outermost wall of the labyrinth heading West and South. The centre of the labyrinth is a square consisting of four square units and becomes the destination point of the mouse robot (designated as T). This cell has only one entrance (way).

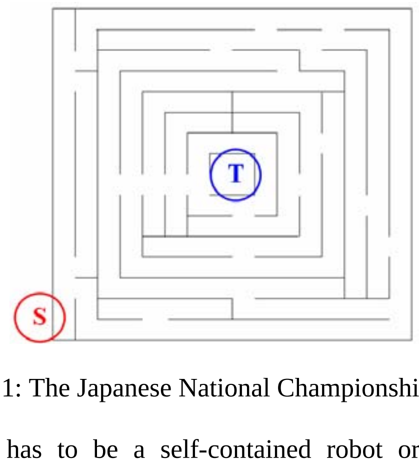

Figure 1: The Japanese National Championship layout.

The robot has to be a self-contained robot or it is not a remotely controlled robot. The robot is prohibited to leave anything behind during its journey in the labyrinth. The robot is also not allowed to jump, climb, scratch, break or destroy the walls of labyrinth. The robot's dimensions are limited by the cell size and must not exceed 25 cm x 25 cm.

The time needed for travelling from the starting point to the destination point is called *run time*. The total time for each run is from the point that the robot is activated for the first time until the distance is measured; this is called *maze time* or *search time*. If the robot needs manual assistance at any time during the contest, then the robot is considered to be in touch. The scoring judgement is based on these three parameters.

Each robot has maximum time of only 10 minutes to explore the labyrinth. Scoring for the micromouse robot consists of handicapped time calculation for each run. Handicapped time is measured from the run time, search penalty and touch penalty.

DESIGN OF THE SEARCHING ALGORITHM

Potential value is any value possessed by any cell (the square area) that represents the minimum distance in numbered rectangular steps from the destination point located at the centre of the labyrinth. Therefore, the destination point will certainly have zero value (0). The potential value of each cell is shown in Figure 2, along with a generated path.

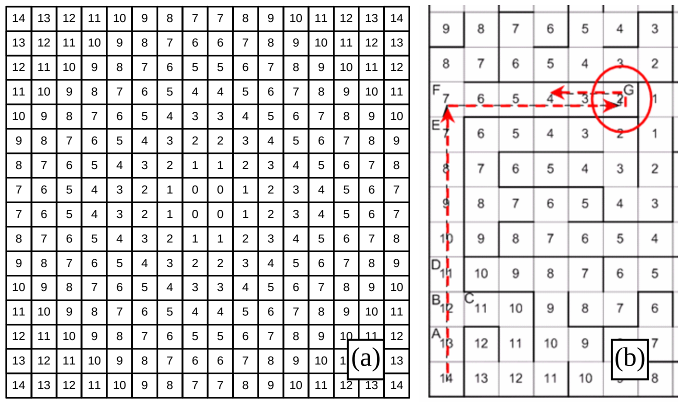

Figure 2: (a) The potential value for each cell and (b) the generated path based on the potential values.

Heading defines the direction or orientation of the robot when it is moving. There are four headings used to define it, ie west, south, east and north. Other variables used in designing the algorithm are as follows:

- *Flag*: a variable indicating if the cell has been visited (false if cell has not been visited and true otherwise);
- *Marking*: a variable indicating prohibition, ie if a cell can be visited or not. Any cell should not be visited if the marking value is true, with three elements setting the marking value to be true: the cell is a dead end (no way out); the cell is a part of dead end; or the cell is part of a closed looping path;
- *Count*: a value identifying how many times the cell has been visited; after the robot arrives at a certain cell, the count variable of this cell is increased by one;
- *Way*: shows the number of alternative ways that can be chosen by the robot. The possible situations are: no way (dead end); one choice, two choices or three choices. This becomes the criteria in making a decision to define which direction should be taken.

When the robot resides in a certain cell, it will perform the following steps:

- First, check whether or not the cell has been visited;
- Check where the walls are;
- Calculate the number of ways;
- Make a decision based on the available number of ways;
- Move between cells, check for closed loop paths and dead ends, and update cell variables;
- Finally, check if the destination cell has been reached.

Strategy for Effective Movement

The main behaviour of the micromouse robot is the capability to make a decision in difficult situations, as follows:

- Execute a dead end procedure (Way=0) when the robot faces three walls as shown at point G in Figure 2(b). Cell G is then marked true and the robot turns around 180°;
- Execute a 1-way procedure (Way=1) when there is only one possible way to go, as depicted in Figure 2(b) from the starting point to A, from D until E, and from G to one cell before F;
- Execute a more_than_1-way procedure (Way=2 or 3), when there is more than one way to go. At cell F, more_than_1-way procedure is accomplished where there are two ways. The robot turns right as the right side cell has a lower potential value (6), which is smaller then the other cell (value of 8).

All of these procedures, ie dead end, 1-way, and more_than_1-way procedures, are equipped with some criteria in making a decision so that the robot can determine which is the best way or which is the next cell to be visited. If the robot has two or three alternative ways, then the first criteria that applies is the *flag* of the neighbouring cell. A flag is a sign given to indicate whether the cell has been visited before. The first possibility is that at least one neighbouring cell has not been passed. In this situation, the robot will choose the unvisited path (cell). But if there is more than one unvisited cell, then the decision is based on the potential value. Potential value criteria is a method to choose alternative paths by selecting the neighbouring cell with the smallest potential value. If there are two cells having the smallest value, then a straight path is selected. Forward movement (path) is set to have a higher priority than turning left or right. This is achieved by considering that moving forward needs less effort than turning the robot. The lower priority is, of course, turning right and then left.

Cell Marking Procedure

When facing a dead end point, the robot should mark all the cells that belong to this end point, as shown Figure 3.

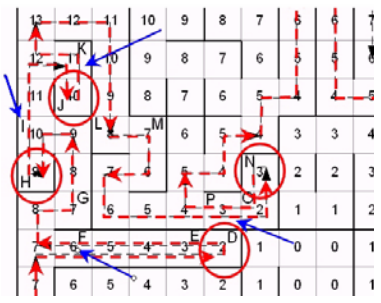

Figure 3: Dead end cell and cell as parts of dead end.

The cells identified to be dead end cells are D, H, J and N. the cells that belong to the dead end are E until F, P and O, and K. Once the cells are marked, then the next time the robot comes to this point, it will not take these points as an alternative way.

Confusing Cells and the Avoidance of Closed Loop Paths

In a certain situation, all neighbouring cells in the path have been visited (Flag=true). In this case, the decision criteria is *count*. The *count* limit is set to 4 as the minimum number to indicate that the robot has come back to the same cell from the same direction. An example is shown in Figure 4, indicating when the robot faces three alternative paths, as follows:

- As the robot moves from A to B, it detects three paths, ie turn left to D, forward to C, or turn right to E. The potential value of E and C are the same. The priority for forward direction is then applied and the robot moves to cell C. The count at B is set to 1 (Figure 4(a));
- When going back to B from A, the count at B is increased to 2. Of the three alternatives, one path has been passed, so it moves to E, which has never been passed and has the smallest potential value (Figure 4(b));
- When returning to B from A, the count at B increases to 3. There is only one possible path left; it then moves to the only cell that has never been passed, ie D (Figure 4(c));
- When the robot returns once more to B, the count at B is set finally to 4 with all alternative paths having been passed (Figure 4(d)).

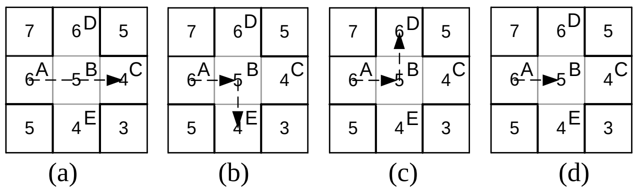

Figure 4: Multiple robot arrivals at the same cell.

Figure 5(a) depicts an example of when the robot faces three alternative paths at point A. From E, it moves to A and the count at cell A is increased by 1, for instance to 4. It then checks the existence of the walls and detects there to be no wall on the left front and right side. It then checks the marking of all the cells and determines that all are false. So the robot has three choices: move to D, turn left to B, or turn right to C. As the count limit (4) is reached, the decision is then based on the count criteria. It is assumed that the count at cell B, C and D are 1, 2 and 2, respectively. The robot then selects the cell with the smallest count, ie cell B.

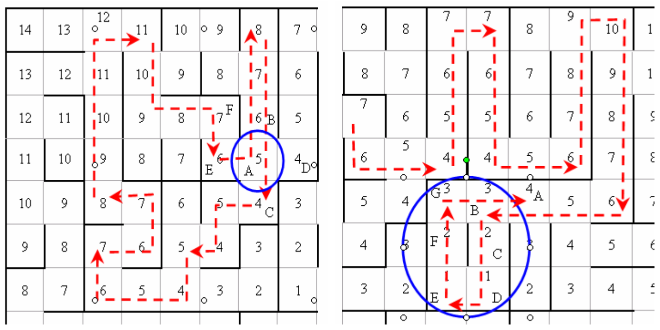

Figure 5: (a) The robot faces three visited paths, and (b) it faces a closed looping path.

In the next trip, the robot goes back to cell A from B. The count at A now becomes 5. All the three paths for the robot, ie straight to C, turn left to D and turn right to E, have been visited. The count of A is already higher than 4. The next step is to check the count of the neighbouring cell. It is determined that the counts are all the same: 2. Based on these values, the next test is to check the potential values of all the neighbouring cells' potential values, with D and C being the smallest numbers. Finally, the robot chooses a straight movement to C based on the forward movement priority.

In some particular places, the robot arrives at a confusing point, where it arrives at the same point again after travelling some distance, as shown in Figure 5(b) and marked by a circle. An algorithm is also developed in order to identify whether a part of the maze forms a closed looping path. In this case, B, C, D, E, F and G are cells that form a closed looping path. When the robot arrives at cell B from cell G, it checks for a second time whether cell B has been visited. As the robot has previously visited cell B, then one previous cell is checked, indicating that cell G is 1-way and the robot can only move to cell B. This checking is repeated for all previous cells until finding a cell that has more than 1-way, ie cell B. Since this cell is the current position of the robot, it concludes that those cells (B, C, D, E, F and G) are part of a closed looping path and the marking is then set to true.

THE DEVELOPMENT OF THE SIMULATION

The algorithm explained above is examined for 10 labyrinth (maze) types in a simulation program developed in *Turbo Pascal* and *Borland Delphi 7* programming languages [8][9]. The 10 labyrinths are the USA, Japan, Canada, Singapore, UK, APEC, ITB, British, Japan2 and TI Unpar mazes. Screenshots of this simulation program is shown in Figure 6(a). The *run* button starts the simulation according to the type of maze chosen in the Maze Menu window. The robot's speed can be adjusted using the track bar by dragging it to the left or right. The *stop* button immediately halts the simulation at the robot's current position, with the left button space showing the total number of cells that the robot has visited. The *close* button terminates and exits the program. Figure 6(b) shows one try when the robot was exploring the TI Unpar maze.

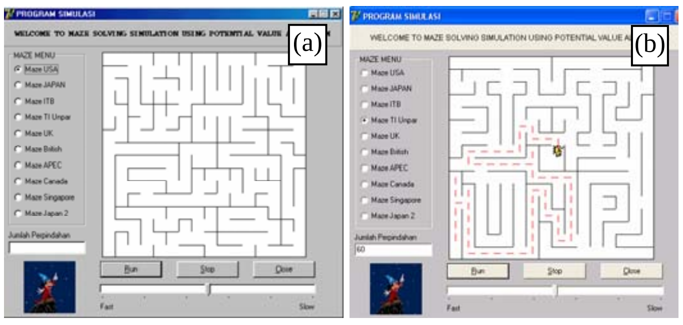

Figure 6: (a) Screenshots of the simulation program and (b) the exploration of the TI Unpar maze.

GENERAL ANALYSIS

To evaluate the algorithm developed in this research, three different algorithms have been executed for the same maze, ie wall following, DFS$^+$ (Depth First Search$^+$) and potential value. The exploration results of the three compared algorithms are shown in Figures 7, 8(a) and 8(b).

In Figure 7, one can see that when the robot implements the wall following algorithm, it fails to reach the destination point.

Conversely, the DFS$^+$ and potential value algorithms lead the robot to successfully reach the destination point. The exploration result of a certain maze shows that using the DFS$^+$ algorithm results in the robot travelling longer than if the potential value algorithm were used. When the ITB maze is chosen, the robot travelling distance using the DFS$^+$ algorithm is 71 times the movement to reach the goal, but when using the potential value algorithm, the robot moves only 41 times.

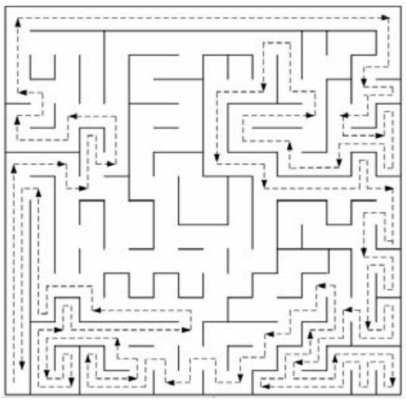
Figure 7: Exploration utilising the wall-following algorithm.

The route passed by the robot using both algorithms is still the same until the fourth cross section (denoted by the circle in Figure 8(a) and the circle in Figure 8(b)). This is caused by the fact that both algorithms make the same decision in approaching the goal. In the DFS$^+$ algorithm, the goal is a labyrinth having the highest level, while for the potential value algorithm, the goal is a cell that has the lowest value. At the fourth cross section, different decisions are taken. Using the DFS$^+$ algorithm, the robot faces an equal option condition, (equal labyrinth level=3) and the priority set is to turn right. This situation is handled differently in the potential value algorithm. The robot turns left as the potential value of the left cell is smaller than the value of the right cell. The results of the travelling distance are tabulated in Table 1.

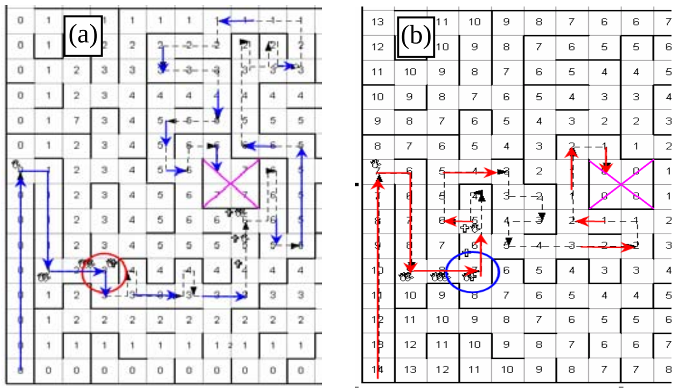

Figure 8: An exploration of the ITB maze using: (a) the DFS$^+$ algorithm and (b) the potential value algorithm.

For the Japan, Canada, ITB, TI Unpar and APEC mazes, the count limit has no effect on the travelling distance. In contrast, the USA, Singapore, UK, British and Japan2 mazes' different count limits correspond to different travelling distances. The USA and Singapore labyrinths have a similar pattern, ie the higher the limit, the shorter the travelling distance. The opposite is the case for the UK labyrinth, where a higher count limit makes for longer travelling distances. The British and Japan2 labyrinths show hyperbolic curves with opposing directions to each other. In conclusion, the optimal count limit depends highly on the type of labyrinth, while the labyrinth itself has not been mapped or is unknown for the robot.

Table 1: A comparison of the travelling distances for the different labyrinth types.

Count Limit		2	3	4	5	6	7
Maze type	Japan	62	62	62	62	62	62
	Canada	186	186	186	186	186	186
	ITB	41	41	41	41	41	41
	TI Unp.	60	60	60	60	60	60
	APEC	64	64	64	64	64	64
	USA	335	335	249	217	217	217
	Singapore	315	307	293	195	195	195
	UK	367	367	617	921	995	1349
	British	320	294	264	272	280	288
	Japan2	342	358	756	962	414	462

Figure 6(a) depicts the exploration in the TI Unpar maze. It needs 60 times movements to reach the destination point. The robot nicely reaches the target, while using the other two algorithms leads to different phenomena. The wall following algorithm fails to perform the task, while the DFS$^+$ makes for longer travelling distances.

CONCLUSIONS

Some conclusions drawn from this exercise are described as follows:

- The searching algorithm for unmapped labyrinth consists of several key steps: checking of the cell status and existence of walls; calculation of the number of ways and the making of a decision based on the available number of paths; movement between cells while also checking whether the destination cell has been reached or not;
- The simulation program has been developed to examine the potential value algorithm and has been executed for 10 labyrinth types;
- The potential value algorithm has successfully reached the target for the 10 types of labyrinth;
- The algorithm developed here is capable of determining a dead end, part of a dead end and closed loop paths;
- The algorithm has been built to prevent the robot from becoming trapped in a closed loop path.

REFERENCES

1. Micromouse UK (2004), http://micromouse.cs.rhul.ac.uk/
2. MicroMouseInfo.com, http://www.micromouseinfo.com/
3. University of California, Berkeley, IEEE Student Branch (2006), http://ucsee.eecs.berkeley.edu/
4. Dr Robin Sarah Bradbeer (2006), http://www.ee.cityu.edu.hk/~rtbrad/
5. Mutijarsa, K. et al, Membangun robot tikus cerdas. *Proc. Conf. on the World of Automation*, Bandung, Indonesian, A-1-40 (2003) (in Indonesian).
6. Micromouse Information Centre Competition Rules (2004), http://micromouse.cannock.ac.uk/rules.htm
7. UK Micromouse Championship Rules http://www.tic.ac.uk/micromouse/toh.asp
8. Borodich, Y. and Leonenko, V., *The Revolutionary Guide to Turbo Pascal*. Birmingham: WROX Press (1992).
9. Guldner J. et al, Multiple wave propagation for global path planning. *Proc. 4th Inter. Conf. on Intelligent Autonomous Systems*, Karlsruhe, Germany, 427-434 (1995).