A New Method for a Micromouse to Find its Way Through a Maze Unaided

Chutisant Kerdvibulvech
Department of Information and Computer Technology,
Rangsit University
52/347 Muang-Ake, Paholyothin Rd. Lak-Hok, Pathum
Thani 12000, Thailand
chutisant.k@rsu.ac.th

Andrew Junmee
Department of Information and Computer Technology,
Rangsit University
52/347 Muang-Ake, Paholyothin Rd. Lak-Hok, Pathum
Thani 12000, Thailand
andrew.j@rsu.ac.th

***Abstract*— A Micromouse is a robot that is designed to find its way through any maze with a black line on a white map. The white map acts as a white background for the maze. Since its external environment is not known, the robot requires some intelligence or decision-making capabilities to find its way through the maze. Therefore, this paper covers on the "decision making algorithm". It is an electro-mechanical robot that consists of 3 main sub systems: the drive system, an array of sensors, and the control system. This paper also covers algorithm and programming logic to solve the maze. Then gradually improves the algorithm to accurately solve the maze on a one-way path with extra intelligence. Our aim is to allow the Micro mouse to reach the end of the maze.**

Keywords—robotic; maze; tracking; decision making; micro-mouse

I. INTRODUCTION

Micromouse is a robot designed to get to the end of the maze, unaided. The robot essentially comprises of a drive motor, steering and turning method to move the robot. It has sensors to detect the lines on the maze and control logic to control the activity of the robot and keep it on track. The robot is powered by a battery and has to find its way from a predetermined starting point to the end of the maze unaided. The shorter the time period it takes, the better the decision-making algorithm. The robot has to keep track of where it is and keep exploring the maze lines and detect once it reaches the finishing point.

The mice is completely autonomous robots that must find its way [1] from a predetermined starting position to the end of the maze unaided. The mouse will need to keep track of where it is and detect when it has reached the goal [2].

Autonomous robots have wide reaching applications, as reviewed in [9], from Bomb sniffing to finding humans in wreckage to home automation. It is possible that a Micromouse can save lives and can be argued that not only saving a life of human but also saving the world. This is why we have contributed to researches and advanced robotic studies. It is important to know the potential the Micromouse has and to use this technology in your own desire.

Several related works have been presented in some Micromouses competing against other micromouse in competitions because of their high-tech Micromoues and advanced technology. [7] The MightyMouse objective is to find the center of a 16 by 16 cell maze for some time, the MightMouse will attempt to make its fastest run from the starting point to the destination cells. We aim to make a maze 2-squared meter long and a robot that can track lines. The difference is that, the maze consists of one-way path from start to finish so finding the fastest run isn't necessary. The maze does not need to be complicated with huge amount of twists and turn.

Nowadays, Micromoues are built to solve mazes. There are still mobile robots. Instead of making the robot autonomous, their idea is to make the maze autonomous so that the mobile robot can reach its goal. The basic algorithm [8] is improved by retaining information of the number of decisions that have been made. Consideration is given to implementation using a limited power microprocessor. Their strategy and technique is the opposite of most Micromouse and significance in a way. But we prefer having the robot guide its way through the maze rather than the maze do it.

Fast vision-based object and person tracking is important for various applications in mobile robotics and Human Robot Interaction [3]. Comparing both experiments together, we noticed that there are many ways in Robotic Tracking. We are mainly familiar with Object Tracker for Robotic Application and Line Following Algorithm. It is good to have a tracker for Robotic Application on your electronic devices, but in practical work this doesn't exist. In the end, you will have to use The Line Following Algorithm. Using these results, it is said that robotic tracking relies deeply on what you tracking for. With more and more experiments taking place we are well aware there will be new and different ways for tracking using robotic.

That autonomous robot could move freely and makes decisions, without any lines, as presented in [4]. Ants show an incredible ability to collectively transport complex irregular-shaped objects with seemingly simple coordination A robot that can coordinate itself on invisible lines relies on effective transportation would be outstanding. But the truth is, this is just a theory. If this theory was to hit the markets, you would be looking at a robotic ant. Looking at their experiments, it is quite possible to achieve. In the coordination phase, you are talking about tons and tons of research and lab work. They aim to go far ahead of any kind Line Following Algorithm and Maze-Solving Algorithm. Having said that, they are basally giving eyes to the robot. But is it worth it? Or would not you want to track something that is achievable today.

Kilobot is deigned to make testing collective algorithms of robots accessible to researchers [5]. In the past, testing collective algorithms was important because without algorithms the robots could not function. The Kilobot gets rid of human errors and generates algorithms in fast and accurate way. This kind of work would have speeded up the amount time we wasted on testing codes and algorithms. Whereas our work compiles a entirely different algorithm than the Kilobot. Moreover, it shows that robotics can be used in various ways depending on the algorithm. The Kilobot uses the algorithm to test algorithms and the Micromouse uses algorithms to track lines.

The kit includes the entire maze solving software and many tips and hints on assembling the Micromouse [7]. Also includes photos and videos of the completed Micromouse. This paper only helps assembly your own Micromouse, but it does not mean it is useless. In fact, it quiet useful to have this knowledge. The problem is you need to know the algorithms to successfully complete a Micromouse. It shows step-by-step how to assembly your robot. The equipment they used a far different than the kit we used [10]. The turn over board has more voltage converters. We guess having more converters your Micromouse will have more power, less power spikes and you won't have to change boards for the next 3-4 months. It is safe to say there are several ways to make a Micromouse but having more voltage converters is not needed.

In their works, a team of students put together a Micromouse that can track and detect walls on 3D dimensional maze. The Micromouse must be able to calculate the width and lengths of the each wall of the entire maze. So the robot can move swiftly and quickly through the maze. [11] Micromouse is an annual competition, held the IEE Region 6 spring meeting, in which an autonomous robot must find its way to the center of a maze. The Line Following Algorithm would not be possible on a 3D dimensional maze with walls and objects. The Line Following Algorithm needs to have a darker line and a brighter background, since the Maze-Solving Algorithm does not track lines. It is completely impossible. The difference is not only on the algorithm but also on the maze. The Line Following Algorithm has to have a 2D dimensional maze on a flat and plain surface.

The way Richard T. Vannoy [12] written out his source code is outstanding. He really cleaned out so much of the useless codes that weren't necessary. His technique and style for programming is a lot better. His code pretty much does the same thing but with less codes. For the mechanical part, we admire that we used a steel ball to hold the bearing rods in place. Whereas in our robot we have a rubber ball, which keeps the robot balanced. Richard T. Vannoy's work [15] is quiet similar to ours, looking specifically on the binary code. The trick [13] to getting the robot to move alone the line is to always aim toward the edge of the line.

It is not a Micromouse, the way they are tracking lines is mainly on light. The robot has a more advanced and new technology than the Micromouse. The NXT sensors [7] are more expensive than the ZX-03 sensor and the outcome is that, you will have a much more accurate calibration when detecting light. Another thing is that, the NXT sensor is so accurate it can even track the edge of the lines, rather than just a normal plain line. A robot that uses the NXT sensors will able to turn swiftly on sharp lines without going off track, whereas in the Micromouse the turns have to be quite squared.

The goal of [9] is to build a Micromouse that is fast enough to compete in Micromouse competitions. The motors and chassis of the robot are faster and wider. The motors allow the wheels to spin faster and the chassis has more space for the robot rotate around the maze without losing any speed. We aim to use the Micomouse to complete the maze with a decent time. Speed is not an issue. Considering, calculating speed and adjusting the motors to speed up your robot is a difficult factor, we do not see the point in doing so. As long as the robot can guide its way through the maze unaided states the fact the Micromouse is a successful project.

Richard's maze [12] is fairly simple, yet it has more turns than on our maze. It would probably take less time to complete the maze on Richard's project. Designing a maze-solving robot, you have to think two steps ahead. For example, the left hand rule and the right hand rule. On a simple maze, there are only 8 possible loops that the robot can encounter. Left Turn Only, Right Turn Only, Left or Right ("T"), Four Way, Straight or Left, Straight or Right, End of Maze, and Dead End. These possibilities are set into the left and right hand rule as data and are stored into the microcontroller. Once the robot is placed onto the lines, it triggers the sensors to send signals to the microcontroller. In our work, there are 3 only possibilities. Left Turn Only, Right Turn Only and, End of Maze. Considering, our maze is one-way path 8 possibilities is not needed. The idea was to complete the maze with less possible. As you can, we taken away 5 possibilities and compiled it to the Micromouse.

RoboMind [16], a robot simulation environment, was presented to solve simply connected mazes. RoboMind is built by software they created. The software is used for testing the robot, so the robot will always find the exit in any maze. The software has various kinds of options and quiet user-friendly, so you don't have to write tons and tons of code, most of it is generated for you. Whereas in our work most of the testing came from practical work, finding errors was less precise. The RoboMind software is good tool to improve testing. The software has an augment reality maze and robot, where users can reorganize the maze-solving algorithm without having to touch the robot in the real world. Testing robots without any software is not impossible, but it is not an easy job. While testing, we had to constantly change batteries over and over again.

II. MECHANICAL DESIGN

There are 9 parts that we used to put together our own Micromouse, which are Microcontroller Circuit, Robot Driving Motor, Cable Line, Input Circuit, Robot Wheel, Robot Rubber Wheel, Standard Grid Plate, Standard Plastic Bar, Light-

Emitting Diode. Once the robot was fully assembled. Codes, algorithms, and testing are needed to place it on the maze.

A. *Overview of robot*

Figure 1 shows the brain of the robot, because this is where all the data is stored. It connects the Robot driving motors, Input circuit and LED to the Microcontroller Circuit. The wires that are attached to the Microcontroller Circuit are used to send and receive information. Under the Microcontroller Circuit a battery pack is attached to it, that gives power to the driving motors.

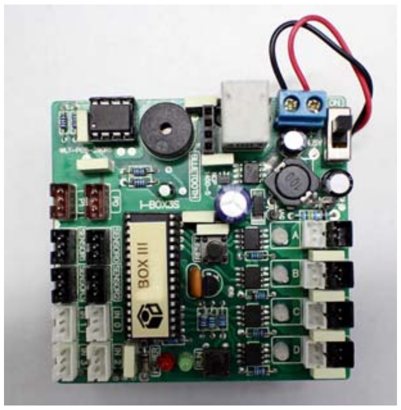

Fig. 1: Microcontroller Circuit

Figure 2 represents the Driving Motor. There are two motors that are attached to the robot wheels, one by one. Driving Motors are used to make the wheels spin and for steering, so that the robot can move forward. It converts any form of energy into mechanical energy and power.

Fig. 2: Driving motor (2 sets)

Figure 3 shows the IR sensors of the robot; they are used like **human instincts**. If there is a turn on the maze, the robot will turn without having to stop. The sensors will feed input to the main circuit then produce an output. By moving left and right, the sensors are tracking whether the line in front of the robot is straight or if it's a turn. Therefore the sensors act more accurately to making decisions. A binary code is then sent to the main circuit as input and output to rotate the robot to make the robot turn. The robot is able to find it's way from the start of the maze to the end, without any errors. There are two wheels on the robot used to balance the robot and of course to make it move.

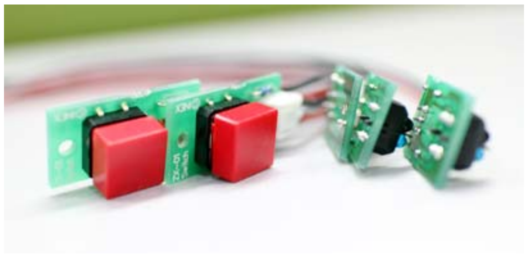

Fig. 3: IR Sensors (2 sets)

Output circuit used to receive input from the main circuit after process and using that output to send to the motors. LED lighting is the eyes of the robot. In an environment with dim lighting the LED lighting is able to detect dark colours and light colours. By doing so, a room with dim lighting is best suited for this kind of Micro mouse, since the dim lighting makes the dark colour visible to the robots sensors.

The programming language we used to write the coding for the Micro mouse is written all in C++. We found it very helpful when applying this language to our robot. It helped us define analog inputs and time limits accurately.

B. *Tracking and The Maze*

In order to execute the algorithm accurately and prevent the robot from going off track the robot has to detect the colour it is moving in. There are major considerations on the design of the robot since various approaches can be introduced in the way the robot sees its environment. One elaborate is that the Micro mouse is color sensitive.

C++ is used to program the microcontroller and the line labyrinth performs the line following algorithm. The robot is able to detect the lines with help of the IR sensors. IR led sends light to the ground, if the ground is black the light is absorbed and the reflected light from the ground will not trigger the transistor. The output of the transistor is 0 when the robot is on the black line. When the robot is on the white background, the reflected IR light is enough to trigger the transistor.

Solving mazes has been studied for years and years in mathematics, which means there are too many algorithms to name here. An algorithm is a set of instructions that, when carried out, produce a certain result. The result in our case is a route to the target destination of the maze.

The maze is made out of plastic. The size of the maze is 2-squared meter long. The maze has a single through-route with twist and turns and is not designed to be as difficult to navigate for the Micro mouse. But there are still some challenges and some will be easier then others.

While the speed of the robot is fixed, it means that the robot is bound to have slightly more problems on twist and turns rather than on a straight line. This is because the robot has to detect where it's position and keep up with how fast the robot is moving. The speed of the robot determines whether the detection is correct and leads to whether the robot will stay on track. Both detection and speed has to be properly adjusted and tested so that both detection and speed meets it making point.

C. *Decision-making Algorithm and P Control*

In every decision-making algorithm consists of a start and end. This algorithm tells the robot, why should we do this? When should we do this? Will it be a success?

Now, these simple questions are answered using a binary code. A binary code represents computer processor instructions using the binary number system's two binary digits, 0 and 1, which only a computer can read.

The robot will always rotate left and right. This is because of the decision-making algorithm. The reason the robot will always rotate left and right no matter what is because the robot is detecting whether the line under the sensor is black or white. So, setting the timer in the decision-making algorithm, we assigned the left steer to 0 and the right steer to 1.

For example a straight line will be written like this 01010101. In binary a turn will be written like this 0010101 or 1101010, which means the first two digits of 0's makes a left turn and carried on straight the first two digits of 1's makes a right turn and carried on straight.

Here is the code for the decision-making algorithm.

```
    if(bot_mode == MODE_ROTATE_R){    //Rotate the robot right (clockwise)...
      track = clip_pos(track);
      r_speed = addsat(REV1,track);
      l_speed  = addsat(FWD1, 0 - track);

      if(timer == 1){             //...until front sensors find a line
        if(motors_stopped()){
          bot_mode=MODE_TRACKLINE_RAMP_UP;
          timer = abs(speed);
        }
 if(bot_mode == MODE_ROTATE_R){    //Rotate the robot left (clockwise)...
      track = clip_pos(track);
      r_speed = addsat(REV1,track);
      l_speed  = addsat(FWD1, 0 - track);

      if(timer == 0){             //...until front sensors find a line
        if(motors_stopped()){
          bot_mode=MODE_TRACKLINE_RAMP_UP;
          timer = abs(speed);
        }
```

P stands for proportional. The idea behind P Control is to have an output, in this case our motors, and a feedback input, our IR sensors, and use the data from the feedback to change the output value accordingly and proportionally.

Tracking light is a major problem in calibration. When tracking a light, you have to think of as it like a mirror. A mirror reflects light in many ways, but we don't want the light to see the robot. We want the robot to see the light. Even if a single light ray can make the robot go off track.

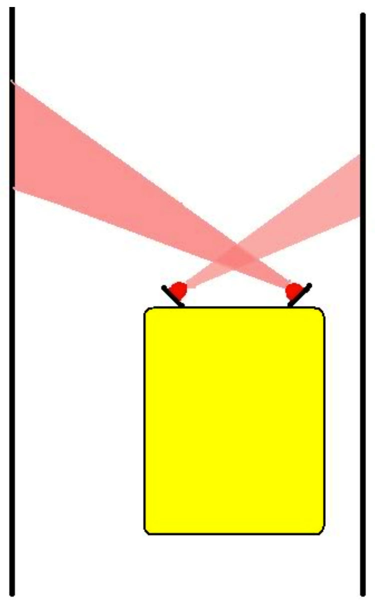

Fig. 4: A scenario where the robot is on the right of the maze.

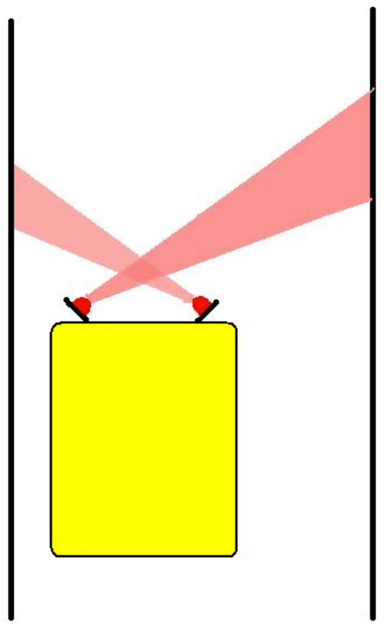

Fig. 5: A scenario where the robot is on the left of the maze.

First, we tried using only two Light sensors, as depicted in Figure 4. One for each side is right and left. But there was still a slight problem with the calibration. So, we added two more

Light-Emitting Diode to the Micro mouse that made it easier and accurate to track light. If the robot is placed on the left of the maze the output of the left-looking sensor will be less than the output of the right looking sensor because it is further away from the maze (IR sensors decrease their output when the object goes away.)

III. Experimental Results

The previous map we applied to the robot was fairly small and the details were poor. We replaced the previous map with a much more challenging maze. To prove the maze works better, we recorded a video of the performance. In the video, the maze has a few dents, which may cause the robot to go off track. We place the robot on the starting point. While the robot is traveling throughout the maze, we are assisting the robot by flatting out the dents. Flatting out the dents supports the robot in reaching the finishing point. A flat surface is the best possible way for the Micromouse to proceed to the finishing point without any errors.

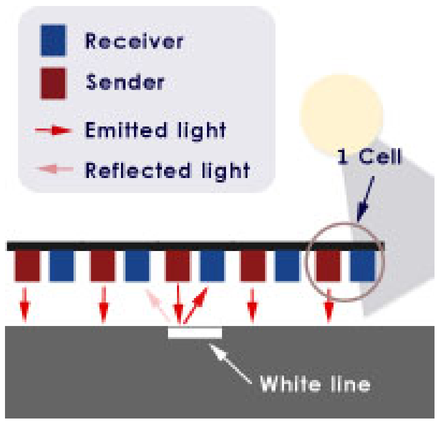

Fig 4. Line-Following Algorithm

The Line Following Algorithm has proven that any Micromouse can be completely robustly. It's a method by which a robot navigates from one position to another by following a line. This line is usually a white line against a darker background or a black line against a brighter background.

As shown in Figure 4, the Line Following Algorithm allows the Micromouse to aid its way through the maze without any assistant just by following the lines in its path. It means, a user does not need a controller. All he/she has to do is push the start button and watch it do its job.

The Line Following Algorithm uses a 2D dimensional maze and contains a flat surface. It has a black line and a white background. It takes approximately 2:30 minutes to complete the maze from the start. The Maze-Solving Algorithm uses 3D dimensional and contains objects and walls, but no lines.

The main reason we decide to make a Micromouse with The Line Following Algorithm is mainly because of its low cost. The Micromouse Kit for the Maze-Solving Algorithm is quiet costly. The way we put together the Micromouse is mostly made out of plastic and Lego parts to stick together the Micromouse Kit.

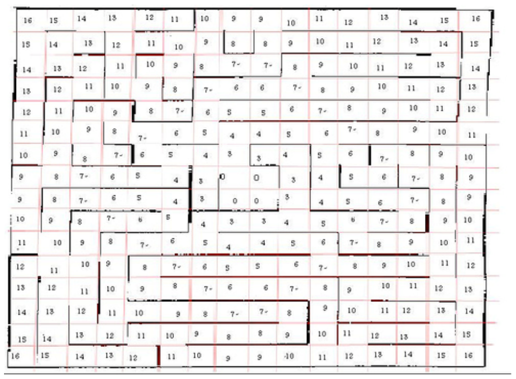

Fig 5. The Maze-Solving Algorithm

For any beginners that want to build a Micromouse we recommend them to go with The Following Algorithm to start with because it is cheaper and the algorithm is easier to learn. Our experimental results can be found online at [6]: http://youtu.be/jWhFa2d_PVA.

IV. Conclusions

The technology as we know is improving everyday [9] and we hope in the near future Micro mouse can be improved in the same way. For instance they are used in many squads around the country. It's intelligent weaponry that can make the world a safer a place.

To the state the fact that the Micro mouse is a success or not is by letting the robot find it's way through the maze without going off track, and without any error detection. In order to keep the Micro mouse on track, it may need some assistants on flattening out the dents on the maze. The problem lays on the maze not the robot. Due to our coding and testing my robot has no problem guiding itself through the maze from start to end. It is proven that our research is safe to say it was a success.

References

[1] http://www.micromouse.ca [website] last retrieved 12/02/2013

[2] http://robogames.net/rules/maze.php [website] last retrieved 12/02/2013

[3] Alexander Kolarow, Michael Brauckmann. "Vision-based Hyper-Real-Time Object Tracker for Robotic Applications"

[4] Michael Rubenstein, Golnaz Habibi, Adrian Cabrera. **"Collective Transport of Complex Objects by Simple Robots"** Theory and Experiments

[5] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. " Kilobot: A Low Cost Scalable Robot System for Collective Behaviours"

[6] http://youtu.be/jWhFa2d_PVA [website] last retrieved 12/07/2013

[7] Kelly Ridge, Sanjeev Giri, Peter Shaw, Jason Flynn. **"MightMouse: An Autonomous Maze Solving Robot"**

[8] L. Wyard-Scott, Q.-H. M. Meng. "A Potential Maze Solving Algorithm for a Micromouse Robot"

[9] http://madan.wordpress.com/2006/07/24/micromouse-maze-solving-algorithm/ [website] last retrieved 29/06/2013

[10] This is a cut down version of the Picaxe Micromouse Manual supplied with the kit.

[11] Minji Kim, IEE UCSD President. **“Micromouse 2011”**

[12] Richard T. Vannoy II, M.S.I.T., B.S.E.E.T. **“Building a Line Following Robot”**

[13] Carnegie Mellon Robotics Academy. **“Line Tracking Basic Lesson”**

[14] Jonathan W. Valvano. “Lab 24 Line Tracking Robot”

[15] Richard T. Vannoy II. “Design a Line Maze Solving Robot”

[16] Research Kitchen, Freshbrain, Hill Air Force Base. **“RoboMind Challenges”**.