Evaluating Recursive Backtracking Depth-First Search Algorithm in Unknown Search Space for Self-learning Path Finding Robot

T. H. Lim(✉) and Pei Ling Ng

Universiti Teknologi Brunei, Tungku Highway, Gadong, Brunei Darussalam
lim.tiong.hoo@utb.edu.bn

Abstract. Various path or route solving algorithms have been widely researched for the last 30 years. It has been applied in many different robotic systems such as bomb sniffing robots, path exploration and search rescue operation. For instance, an autonomous robot has been used to locate and assist a person trapped in the jungle or building to exit. Today, numerous maze solving algorithms have been proposed based on the some information available regarding the maze or remotely control. In real scenario, a robot is usually placed in an unknown environment. It is required for the robot to learn the path, and exhibit a good decision making capability in order to navigate the path successfully without human' assistance. In this project, an Artificial Intelligence (AI) based algorithm called Recursive Backtracking Depth First Search (RBDS) is proposed to explore a maze to reach a target location, and to take the shortest route back to the start position. Due to the limited energy and processing resource, a simple search tree algorithm has been proposed. The proposed algorithm has been evaluated in a robot that has the capability to keep track of the path taken while trying to calculate the optimum path by eliminating unwanted path using Cul-de-Sac technique. Experimental results have shown that the proposed algorithm can solve different mazes. The robot has also shown the capability to learn and remember the path taken, to return to the start and back to target area successfully.

Keywords: Maze solving · Search space · Graph theory · Search tree · Depth first search · Cul-de-sac

1 Introduction

Technology advancements in the area of robotics have made enormous contributions in both industrial and social domains. The applications of automation and robotics have increased significantly from home to industrial automations, bomb sniffing robot, and search rescue operation [1,2]. For such applications to operate in a dynamic environment, the systems must exhibit autonomy and decision making properties. In a fully autonomous robot, it is important that it can perceive and adapt to the environment, and operate for an extended period without human intervention. After operating for a period of time, it should learn about its new environment and become more intelligent. In the perspective of intelligence in a robot, it can be derived from the fields of Artificial Intelligence (AI) that compromises of three separate entities namely search algorithm, knowledge representation and the extent of implementing these ideas.

In the application of search and rescue, speed and efficiency is much more concerned due to the urgency to locate the position of wounded or trapped persons in order to save them [3]. These complex situation of accident locale can be abstracted using a maze. Therefore, to test the intelligence, these robots are required to solve problem such as solving a maze. A maze is network of paths and hedges designed as a puzzle through which solver has to find a way or solution, usually from an entrance to a goal or another exit. In general, there are two types of mazes [4] namely model-based maze in which global model is available, and sensor information-based maze of which information about the maze is unknown.

Many algorithms for maze navigation and maze solving have been proposed and continue to be improved over the years [3,5–7]. Artificial Intelligence plays a vital role in defining the best possible method of solving maze effectively, such that Graph Theory appears to be an efficient tool while designing proficient maze solving techniques [8]. Primarily, the search problem includes either an explicit initial or goal node in which one has to find the path connecting them or simply find the goal which is implicitly defined [9].

It is fundamental for the robot to successfully navigate to its goal. This is usually achieved either with direct human navigation, a predefined program or set of general rules programmed in the systems. These approaches usually assume that global model or view is available in advance [7]. This assumption may not be always true as the environment to be traversed is usually unknown in advanced. Each maze is usually different and may change dynamically. Hence, the use of fixed rules or predefined path is not the best approach. In order for a robot to traverse through an unknown environment from a source position to a target, it needs to percept and act using current available information. The key information of the maze's search space, situation of branches, dead-end or passing-through crossing and current perception of the robot are all acquired locally from sensors attached to the robot during the search.

In this paper, a new Artificial Intelligence (AI) based algorithm called Recursive Backtracking Depth-first Search (RBDS) is proposed and evaluated in real robots. To the best of our knowledge, this is the first time a search tree based AI algorithm has been applied in a low processing and powered robot. The motivation behind re-using an existing simple known AI algorithm is that we do not want to design a whole new algorithm. In contrast, the simplest search algorithm can easily be applied and what is proposed in this paper can be supported by any microcontroller such as Arduino and microPy. We have also incorporated a route optimization algorithm to allow the robot use the shortest and optimal path to traverse between the goal and initial start point.

The main contributions of this paper are: 1) A novel AI-Based algorithm called Recursive Backtracking Depth-first Searh (RBDS) using an uninformed Depth First Search (DFS) to explore an unknown search space and applied an optimization algorithm called Recursive Cul-de-Sac Backtracking (RCB) to eliminate all unwanted path for route optimisation; 2) A quantitative analysis of the AI-Based RBDS algorithm on a robotic platform based on a low power, and low cost microcontroller robot to solve an unknown maze autonomously. A reinforcement learning technique is used by the robot to learn about the new environment and update its environment. Initially, the robot traverse through an unknown search space using the information received from an input sensor. Once the goal is located, the robot will identify the optimal path by eliminating the dead ends using RCB. In the subsequent run, the robot can traverse between the goal and initial position using the optimal path. The results have shown the proposed algorithm can help to solve any maze and determine the optimum path..

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 provides an overview of AI based RBDS follows by the experimental setup to analyse the performance of the proposed system in Sect. 4. In Sect. 5, we summarize with our conclusions.

2 Related Works

Autonomous mobile robots need to operate safely and reliably. At the same time, they are expected to minimize energy consumption, travel time and distance. Mishra and Bande [10] conducted a comparative study on the path length and time taken to solve the maze using 3 different algorithms namely basic wall follower, Djikstra's shortest path, and flood fill algorithm [11]. Mishra and Bande show that the Djikstra's algorithm can be effectively used if both time and hardware are not critical and supported. However, both constraints need to be satisfied, flood fill would be preferable compared to the others despite the complex calculation required. The basic and simplest wall follower logic is the most supported algorithm in hardware. In another study, Sharma [12] the same results are also obtained where flood fill algorithm outperformed basic wall follower logic.

Another comparative study on wall follower, Lee's algorithm [13] and flood fill algorithm [5] were presented by Gupta and Sehgal [14]. Equally, Lee's algorithm is an application of Dijkstra's algorithm that uses Breadth First Search (BFS). It works in two phases namely the filling phase in which cells are marked while retrace phase is derived from the concept of backtracking [13]. In BFS, the tree is examined from top down such that every node at depth d is examined before any node at depth d+1. Starting from the source vertex, the entire layer of unvisited vertex is visited by some vertex in the visited set and the recently visited vertexes are added with the visited set [15]. Although floodfill is by far the most famous and efficient algorithm to solve all types of maze, it is commonly set with a preset target point. Initially, the algorithm assigns to a value of each cell representing the distance between the cell and the target cell. The robot follows the path of decreasing value and is updated in every single step.

Finally, in Depth First Search (DFS), the search starts from the root of a graph. The terminal nodes are examined from left to right. The exploration continues toward the deeper region of the tree or graph until it reach a dead-end and needs to back track. The algorithm starts searching from a specific vertex and then it navigate by branching out corresponding vertices until it reach the final goal [16]. The whole maze is mapped as graph where the nodes or vertexes are considered as maze cells. The search order may not be in specific order, it only commits the idea of always expanding a node as deep in the search tree as possible. Hence a search from right to left is also possible.

3 AI-Based Recursive Backtracking Depth-First Search (RBDS) Algorithm

In this section, an overview of design and methodology used for Artificial Intelligence Based Maze Solving robot is presented. This includes the construction and implementation of two-wheel robot that is capable of solving a line maze using an array of infra-red sensors. A line maze is used instead of the real wall maze as it is cheaper to built and modify. The robot will be used to evaluate the propose Artificial Intelligence (AI) based algorithm known as Recursive Backtracking Depth First Search (RBDS), capable of searching through an unknown search space to identify a target without any guidance.

3.1 Systems Design

The system design of the autonomous robot consists of three fundamental phases in the design process. The first involved the selection and integration of individual components of a mobile robot to operate according to the system specification required to traverse the maze. Secondly, the design of the RBDS algorithm and apply Cul-de-Sac approach to elimate any deadend from the database in order to optimize the route taken. Finally, the interface the system software with hardware components with the RBDS algorithm in order for the evaluate the robot to operate autonomously using RBDS.

The proposed path finding algorithm is based on an uninformed or blind search, Depth First Search algorithm. The operating environment is in the form of a maze with black lines representing the path and having different kinds of junctions. The starting point can be randomized with target goal differentiating from other paths. Figure 1 shows the overall system of block diagram of maze solving robot.

The working principle of robot in solving the line maze is provided by the systems program functions listed as follows;

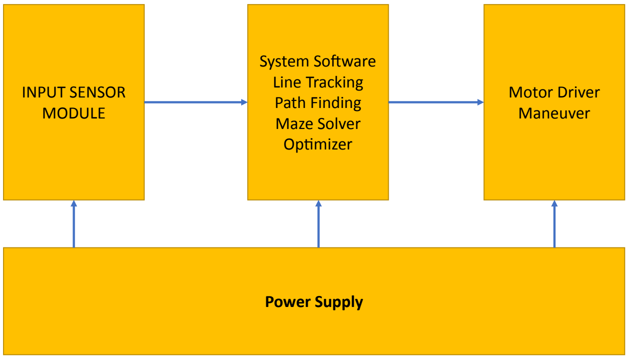


Fig. 1. Robotic systems design.

- Line tracking system In line tracking system, the robot follows a black line maze by sensing the environment using the input sensor module. The sensor values are sent to the software system, in which the microcontroller will process. The program function will instruct the robot to follow the line as it percepts and acts accordingly to the instruction sent from the system to the motor drive system.
- Depth First Search Based on the output sensory module information, the robot teaches itself to identify the route path taken using path finding algorithm. The robot will apply the Depth First Search Algorithm to explore the maze to achieve its goal and use its knowledge through the computational perception and action condition rules to learn about the route to reach the goal.
- Recursive Backtracking Path Optimisation Function After identifying the route path, the robot enters the learning phase and optimises the path. In subsequently run, the robot returns to its original source position using the optimised path. Once it returned to the original position, the robot can now operate for instance transport and transfer goods from the start to the desired position using the optimised path.

3.2 Platform Design

Figure 2 shows the circuit diagram of hardware system of the robot. It consists of 3 major hardware components namely the processing unit, the input unit and the output unit. An Arduino microcontroller processing unit is used to operate the robot. As the core component of the systems, it is responsible for processing the input data and storing the information produced during the computational process. The input and output units are responsible robot's perception and actions. Input information regarding the line tracks, dead end and turns will be obtained from the five channel infrared reflective sensor array module. The track or movement can be determined by sending an infrared signal to the line while photo-transistor receive and sense the signal correspondingly. The correspond information from the sensors will be fed into the Arduino. Arduino will then process the signals and convert them to digital values with an integrated analog to digital ADC interface. The resulting results will then be compared with the programmed action condition rules to generate appropriate output instruction to the motor drive system. The robot is driven by a motor driver H-bridge IC L298N and 2 DC geared motor. The wheels of the robot are capable of independent rotation in two directions.

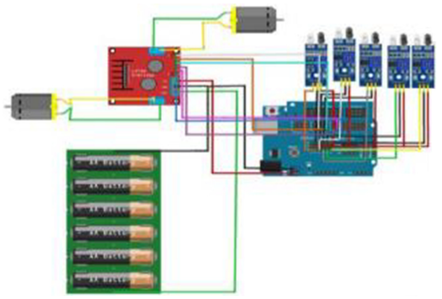


Fig. 2. Hardware circuit design for the autonomous robot

3.3 Software Design

The system application design of the AI based autonomous maze solving robot is shown in Fig. 3. C programming language is used to develop the AI based RBDS algorithm to traverse and solve a maze. The program code acts as the decision maker using Depth First Search Algorithm embedded in the microcontroller. The robot percepts and acts based on the output for particular set of combination of sensors.

The working principle of robot in solving line maze is as follows;

- Line path tracking.
  For line path tracking, the robot follows a line maze by perceiving its environment using array IR sensors. It is programmed to follow the line perfectly with both motors rotating in forward direction. Any deviation from the line will cause the motor change accordingly to ensure it is still on and within the right path. The most right and left sensors are used for taking decision on junctions while the other 3 middle sensors are for tracing the black line, for forwarding.
- Mobile robot navigation.
  For navigation, the robot is dependent on the sensors. The output from the sensor is fed into the Arduino, microcontroller which is programmed based on the action condition rules. As the input is processed, the corresponding output is sent to the H-bridge, L298N motor driver to navigate the robot by rotating it in corresponding directions.

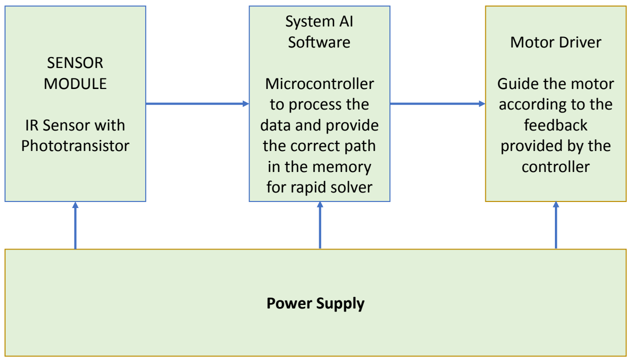

SENSOR MODULE
IR Sensor with Phototransistor
System AI Software
Microcontroller to process the data and provide the correct path in the memory for rapid solver
Motor Driver
Guide the motor according to the feedback provided by the controller
Power Supply

Fig. 3. Software system block diagram

- Knowledge based navigation and reinforced learning.
  The knowledge based navigation is incorporated by means of Depth First Search, left search variant rule. The first priority is given for the left, second to the straight and least to the right. Based on the knowledge, the robot navigates accordingly. In the teaching phase, it uses the rule to navigate and traverse through all paths including dead ends. When it encounter dead end, it takes a back turn or 180° turns and traverses the same path again. The turns are stored only at junctions, intersection, later used by the robot for navigation in the subsequent phase. In the second run, Depth First Search is not applied, instead the Cul-de-Sac technique is applied such that optimised path is used to traverse the robot back to its original position by eliminating unnecessary turn and dead ends. Alternatively, the robot can also go back to the goal from the start position using the optimised path learnt.
  The use of Artificial Intelligence concept is similar to the reinforcement learning applied in the second run. Where it illustrates the concept of learning from mistake. The former teaching phase, enable the learning of how to solve the maze as it proceeds through dead ends. Subsequently the next phase involves Cul-de-Sac approach by eliminating all the dead ends and remove them from the stack memory. As a result, an optimal path to the goal from the start position is identified.

3.4 Algorithms

A Depth First Search, left hand rule with Cul-de-Sac approach is used as the algorithms to solve the line maze. This algorithm is qualitative in nature, requiring no map of environment, no fundamental matrix and no assumption as used in other algorithms.

Line Tracking Depth First Search. Similar to classical strategy for exploring maze, DFS algorithm is used for exploring a maze, The DFS follows the left hand rule such that it commits the idea of always expanding the left hand node and move down in the search tree following the known rule. The left hand rule states the left direction has the highest priority compared to the straight and right direction. Similarly, the straight direction has higher percedence compared to the right. The precedence order is as follows: Left, Straight, Right. At each intersection, junction, and dead end (vertex) visit are marked, so that the path back to the entrance, start vertex could be tracked using recursion stack.

During the first phase, the robot follows the black line path and traverse to find the desired goal. When traversing the maze to find the desired goal, at every junction that requires decision making, the turn the robot takes is recorded and stored in the stack of Arduino. The moves are denoted by S when it move forward, L when it turns left, R when it turns right and B for turning back when it encounters dead end. When the desired goal is reached, the robot finishes the first phase and proceeds to the next phase of learning. The robot will then traverse from the target to the starting point and vice versa though the optimal path. The optimal path can be determined using Cul-de-Sac approach to simplify the path.

Cul-de-Sac Approach. Once the goal node is found, path simplification can be achieved using Cul-de-Sac approach to find the optimal path in the traversed maze. At every turn, the length of the recorded path increases by 1. The algorithm takes the turn stored in the stack during the first phase and applies Cul-de-Sac approach, respective reduction path rules accordingly and removes all the dead end, back turn made previously.

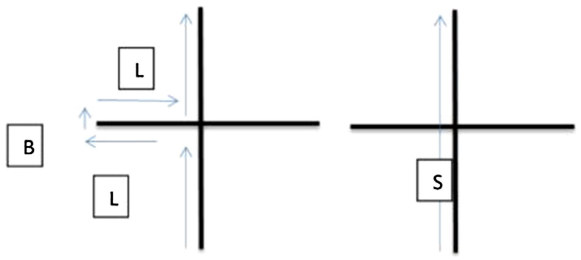


Fig. 4. Correspondence of LBL sequence and equivalent path

With path simplification, the strategy is that whenever the robot encounters a sequence xBx, the path can be simplified by eliminating the dead end and replaced with a turn corresponding to the total angle. The path can only be simplified if the second to the last turn was “B” a dead end and the path length must be greater than 3. Considering the sequence “LBL”, after turning 90° to the left and 180° back and 90° to the left again, the net effect of the robot is heading back in its original direction. The path can be simplified to 0° turn, a single “S” move.

The Cul-de-Sac algorithm is based on the data in Table 1. In a traversed maze, the algorithm is applied onto the list of path moves stored in an array stack several times until all the necessary "B" moves are removed. Figure 4 exhibits two functionally equivalent paths from the start to the end.

Table 1. Path simplification

Original path sequence	Reduced path
LBL	S
LBS	R
LBR	B
SBL	R
SBR	L
SBS	B
RBR	S
RBL	B
RBS	L
L = left, R = right, B = back, S = straight	

3.5 Robot Driving Control System

To allow the system to drive the robot autonomously, the robot is mounted with five channel infra-red sensors. These five sensors will be used to detect a straight line, turns, dead end or goal and the robot can use the detected information to navigate the robot. With five input values, the sensors can have numerous configuration that allow the robot to make decision whether to move straight or to turn. With digital output, there are 32 possible combinations. With that, '0' output indicates sensor sense black line and '1' output sense otherwise.

As shown in Fig. 5, the far most left and right sensors detect turning and intersection. The three sensors in the middle are evenly spaced. Chance of two sensors detecting line at the same time when it adjust itself is high. Thus, few combinations are accounted for in the action condition rule. These sensors looks directly down on the line track and processed by the program to determine the correct action. The following figures exhibit some of the possible sensor output combinations for following a line.

As the robot is expected to follow the lines and find the path from the source position to the desired goal. There are various junctions in the line maze. Given a maze with no loops, there are only 8 possible scenarios that the robot would encounter. First, it follows the line. When it reaches an intersection, the sensor will decide what type of intersection it is and make appropriate turn. These steps continue in a loop until it reach the maze end.

With the path finding algorithm implemented, the robot acts accordingly as it perceives the environment, depending on the type of turns and intersection it encounters. In the case of left turn junction and right turn junction, the robot has no alternative but to make 90° turn. As of reaching a dead end, the robot make a 180° u-turn to backtrack. For the straight or right turn junction, a sub routine is created. The robot moves forward for a bit and perceive the current state's its in. If the robot sense there is line ahead, it moves forward as the rule prioritizes straight rather than right. As of the left or straight junction, the robot will always prioritize and turn left. The junctions can be identified according to the sensor readings tabulated in Table 2.

Table 2. Truth table For direction and motor rotation

S1	S2	S3	S4	S5	Junction	Rotation
0	0	0	0	0	Left	Left
1	1	0	1	1	Following line	Straight
1	1	0	0	0	Left	Left
1	1	1	0	1	Slight left	Left
1	1	0	0	1	Slight left	Left
1	0	1	1	1	Slight right	Right
1	0	0	1	1	Slight right	Right
0	0	0	1	1	Right	Right
0	0	0	0	1	Slight left	Left
1	0	0	0	0	Slight right	Right
1	1	1	1	1	No line	U-Turn
1	0	0	0	1	End maze	Stop

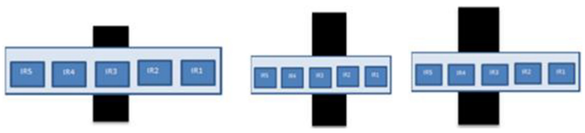
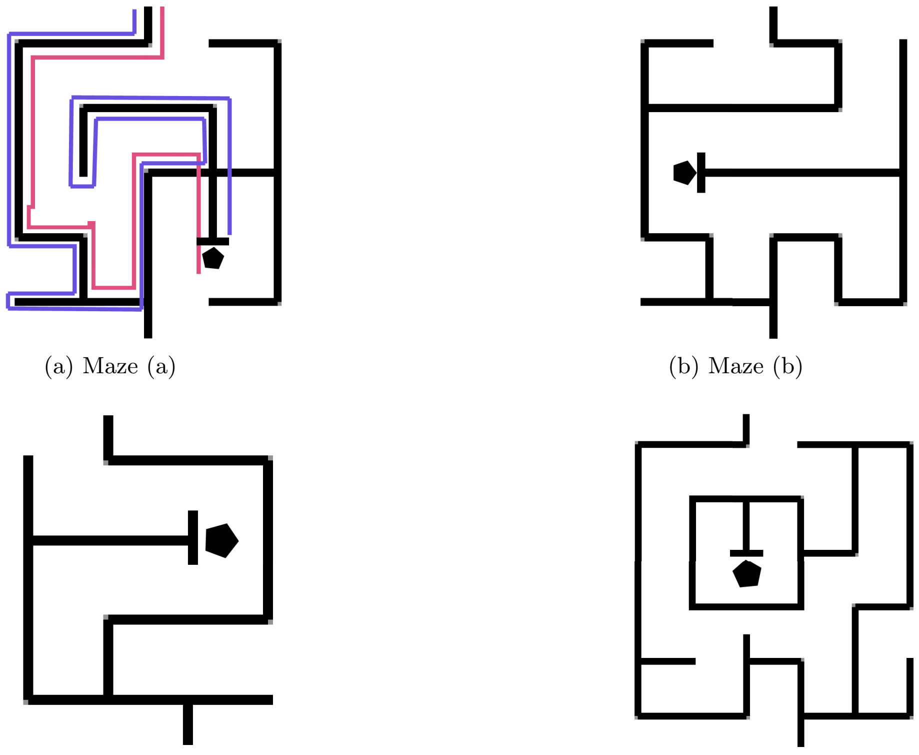


Fig. 5. Sensor combinations for following line

4 Performance Evaluation

For the evaluation of RBDF Search Algorithm, 4 line mazes are tested. Figure 6 shows the corresponding line mazes to be solve. The robot is placed at a defined source position with differentiated desired goal position to test if it is capable of solving the maze autonomously. Alternatively in each line maze with a fixed goal, the robot start position is also chosen randomly.



(a) Maze (a) (b) Maze (b)

(c) Maze (c) (d) Maze (d)

Fig. 6. Evaluation of RBDF on different mazes

4.1 Analysis

From the results obtained and observed, the algorithms are able to solve mazes in the teaching phase and learning phase. In the first phase, the learning phase indicated by the blue line in Maze (a), the robot traverse and store the paths it have taken until it reaches the goal. In the subsequent run indicated by pink line, using Cul-de-Sac approach it backtracks the path to reach the starting source position using an optimal path. Once it reaches the source position, it finishes the run by traversing through the optimal path to the desired goal. For looped maze as shown in Maze (d) Fig. 6, the fourth maze, the algorithm was not able to solve the maze. This is due to the Depth First Search algorithm, Left Hand Rule following such that the robot will always choose to move in the left direction instead of turning right to reach the goal.

5 Conclusion

In this paper, an AI based maze solving robot has been proposed and evaluated on a robotic platform. This autonomous robot can be deployed in a location that is not accessible by rescue team or endangered to human's intervention. It uses RBDS to calculate the optimal path to move from one point to another in the most energy efficient way, thereby reducing the power consumption. With Cul-de-Sac approach, the uses the best or shortest path to go back to the start position without traversing the whole maze. Although the robot evaluation platform can successfully solve the maze autonomously, there are still issues such as power availability and sensitivity of the sensors affecting the route tracing that need to be addressed. Since the design is based on 8 V power supply, problems such as voltage drop due to rapid discharging of battery are observed. Sometimes, it has also affected the tracking of the path and decision making process due to the voltage drop in the sensor-motor driver system. Hence, it is necessary to address this issue before this platform can be used to evaluation the AI algorithm.

References

1. Abdullah, A.H., Lim, T.H.: SmartMATES for medication adherence using non-intrusive wearable sensors. In: Perego, P., Andreoni, G., Rizzo, G. (eds.) MobiHealth 2016. LNICST, vol. 192, pp. 65–70. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58877-3_8
2. Muhammad, N., Lim, T.H., Arifin, N.S.: Non-intrusive wearable health monitoring systems for emotion detection. In: 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 985–989 (2017)
3. Stentz, A.: Optimal and efficient path planning for partially known environments. In: Hebert, M.H., Thorpe, C., Stentz, A. (eds.) Intelligent Unmanned Ground Vehicles. The Springer International Series in Engineering and Computer Science (Robotics: Vision, Manipulation and Sensors), vol. 388, pp. 79–82. Springer, Boston (1997). https://doi.org/10.1007/978-1-4615-6325-9_11
4. Jiang, H.L.: Designed of Wheeled Robot Based on Single Chip Computer, vol. 13 (2009)
5. Elshamarka, I., Saman, A.: Design and implementation of a robot for maze-solving using flood-fill algorithm. Int. J. Comput. Appl. **56**(5), 8–13 (2012)
6. Dang, H., Song, J., Guo, Q.: An efficient algorithm for robot maze-solving. In: 2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics, vol. 2, pp. 79–82 (2010)
7. Cahn, D.F., Phillips, S.R.: ROBNAV: a range-based robot navigation and obstacle avoidance algorithm. IEEE Trans. Syst. Man Cybern. SMC **5**(5), 544–551 (1975)
8. Sadik, A.M.J., Dhali, M.A., Farid, H.M.A.B., Rashid, T.U., Syeed, A.: A comprehensive and comparative study of maze-solving techniques by implementing graph theory. In: International Conference on Artificial Intelligence and Computational Intelligence, vol. 1, pp. 52–56 (2010)
9. Cai, J., Wan, X., Huo, M., Wu, J.: An algorithm of micro mouse maze solving. In: 2010 10th IEEE International Conference on Computer and Information Technology, pp. 1995–2000 (2010)

10. Mishra, S., Bande, P.: Maze Solving Algorithms for Micro Mouse, pp. 86–93, January 2009
11. Wang, H., Yu, Y., Yuan, Q.: Application of Dijkstra algorithm in robot path-planning. In: Second International Conference on Mechanic Automation and Control Engineering, pp. 1067–1069 (2011)
12. Mishra, S., Bande, P.: Maze solving algorithms for micro mouse. In: 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems, pp. 86–93 (2008)
13. Lee, C.Y.: An algorithm for path connections and its applications. IRE Trans. Electron. Comput. EC **10**(3), 346–365 (1961)
14. Gupta, B., Sehgal, S.: Survey on techniques used in autonomous maze solving robot. In: 2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence), pp. 323–328 (2014)
15. Ginsberg, M.: Essentials of Artificial Intelligence. Elsevier Science (2011)
16. Adil, M.J.S., Maruf, A.D., Hasib, M.A.B.F.: A comprehensive and comparative study of maze solving techniques by implementing graph theory. In: International Conference on Artificial Intelligence and Computational Intelligence (2010)
17. Lim, T.H., Lau, H.K., Timmis, J., Bate, I.: Immune-inspired self healing in wireless sensor networks. In: Coello Coello, C.A., Greensmith, J., Krasnogor, N., Liò, P., Nicosia, G., Pavone, M. (eds.) ICARIS 2012. LNCS, vol. 7597, pp. 42–56. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33757-4_4
18. Lim, T.H., Bate, I., Timmis, J.: A self-adaptive fault-tolerant systems for a dependable wireless sensor networks. Des. Automat. Embedded Syst. **18**(3–4), 223–250 (2014)