

## Path discovering in maze area using mobile robot

Ibrahim Majid Mohammed<sup>1</sup>, Mustafa Zuhaer Nayef Al-Dabagh<sup>2</sup>, Salar Jamal Rashid<sup>3</sup>,  
Nor Ashidi Mat Isa<sup>1</sup>

<sup>1</sup>School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Pulau Pinang, Malaysia

<sup>2</sup>Department of Computer Science, College of Science, Knowledge University, Erbil 44001, Iraq

<sup>3</sup>Department of Technical Training, Computer Center, Northern Technical University, Mosul, Iraq

### Article Info

#### Article history:

Received Jan 07, 2021

Revised Jan 17, 2022

Accepted Jan 25, 2022

#### Keywords:

Left-hand algorithm

Maze-solving

Mobile robot

Virtual maze

### ABSTRACT

Robotic maze pathfinding problems deal with detecting the correct route from the start point to the end-point in a virtual maze environment consisting of walls. Automated robot mobility is a significant feature, which enables a mobile robot to traverse a maze independently, from one position to another, without human intervention. There is a myriad of autonomous industrial mobile robot applications, including the transportation of goods and parts, domestic cleaning, indoor security surveillance, airport baggage couriers, and a plethora of other applications to traverse dangerous locations. This paper proposes a pathfinding mobile robot in a virtual maze based on a combination of a simplified left-hand algorithm and a line-following control algorithm. The mobile robot works in any maze to determine a route from the initial starting point to the end-point. The approach outlined in this paper uses a left-hand algorithm to solve the maze problem and a line-follower control algorithm to enable the robot to move in a straight line through the virtual maze. The algorithm used is less complicated and prevents the robot from falling into infinity loops compared to the traditional wall-follower algorithm.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



### Corresponding Author:

Ibrahim Majid Mohammed

School of Electrical and Electronic Engineering, Universiti Sains Malaysia

Pulau Pinang, Malaysia

Email: [ibrahim.m.94@student.usm.my](mailto:ibrahim.m.94@student.usm.my)

## 1. INTRODUCTION

Autonomous movement systems are mobile machines capable of communicating with the environment and performing various tasks without human intervention [1], [2]. In mobile robotics, the robot can be moved from one point to another automatically in a way called autonomous navigation [3]. Despite the massive development in the area of robotics, robots have many difficulties in real environments during navigation because real environments are dynamic, complex, uncertain, and unknown [4].

A maze is a network of trails and borders. A maze is a puzzle in which someone has to discover a way through [5]. Maze-solving algorithms have difficulties dealing with real-world limitations. It is vital to study those limiting factors to correctly design the software for a maze-solving robot [6]. It is easier to program the robot “what”, rather than “how” to do a task [7].

Generally, there are two kinds of maze traversing classifications: 1) a model-based algorithm, whereby the objects are known beforehand; and 2) a sensory information-based or learning algorithm, which recognizes entirely unknown objects. The sensory information-based robot learns its environment by evaluating the data provided by the sensors [8]. There are many kinds of sensors that use in the design of robots such as tactile sensors, visual sensors, and laser sensors.

A robot designed to traverse unknown confined spaces, such as underground tunnels, caves, and pipes, would depend on a limited number of sensors because the robot must be light and small. Typically the robot would sense using small, low power consumption, infrared, cliff, and ultrasonic sensors, as a small, lightweight robot will have a small power source [9], [10]. Maze-solving is the act of discovering away in the maze. Some maze-solving approaches have no prior knowledge of the maze; conversely, in other approaches, an individual can see the complete maze instantly [8], [11].

This work uses Matlab software to create several virtual mazes. The mobile robot utilizes a simplified wall follower left-hand approach to find the correct path from the initial point to the end-point. This proposed algorithm uses only two sensors instead of three, thus decreasing the weight of the robot. Furthermore, the proposed algorithm uses a line follower control algorithm to enable the robot to drive in a straight line through the virtual maze. The organization of the sections of this research paper is as follows: section 2 presents the proposed method for the pathfinding of the mobile robot, section 3 shows the main results of this work in a virtual environment, and the last section presents a summary conclusion of the work.

## 2. MOTIVATION

The problem of solving a maze is a classic robotic problem discussed for more than three decades. However, it is still a fundamentally significant area of robotic systems [12]. Autonomous maze solving is dependent on the robot's decision-making algorithm. After placing the robot in an unknown environment, the robot executes its decision-making algorithm concerning its sensory inputs to achieve its goals. Many maze-solving algorithms are available such as Lee's algorithm, flood fill algorithm, wall follower - to name just a few [13]. The wall follower algorithm typically defines the end-point target with a specific sign. However, the location of the target is initially unknown to the robot. The wall follower algorithms can be implemented based on the right-hand or left-hand rule [14].

The algorithm defined in this research uses the left-hand rule: the decision tree performs sensory tests from left to right. If the robot cannot first turn left, then it will check the forward direction. In the next step, it will attempt a right turn. If it cannot execute any of those movements, the robot will turn right again, having turned 180 degrees, then return to the last successful turning point [15]. Upon returning to the last successful turning point, the robot will be facing the other way, and it can execute the decision tree from left to right, find a different outcome, or return to the next previous point. Similarly, the right-hand algorithm works in the same way, but its decision tree starts from right to left [16]. The left-hand wall follower algorithm does not need the right side sensor because the robot only senses left and forward walls.

### 2.1. The motivation of robot kinematic model

The kinematic model is used to find the robot speed in the inertial frame as a function of the speed of wheels and the robot geometric parameters [17]. Assume  $pp = [xx \ yy \ \theta\theta]^T$  represent the robot posture vector, where  $x$  and  $y$  represent the robot location, and  $\theta$  is the angle between the heading direction and the X-coordinate, as shown in Figure 1. The angular velocities  $\omega_L$ ,  $\omega_R$  control the wheels of the robot. The relationships between the circumferential speeds  $V_R$ ,  $V_L$  and  $\omega_L$ ,  $\omega_R$  are [18]:

$$V_L = R\omega_L \quad (1)$$

$$V_R = R\omega_R \quad (2)$$

where  $R$  represents the radius of the wheels.

These equations can represent a robot, kinematic model:

$$\dot{x} = V \cos(\theta) \quad (3)$$

$$\dot{y} = V \sin(\theta) \quad (4)$$

$$\dot{\theta} = \omega \quad (5)$$

$$V = \frac{V_L + V_R}{2} \quad (6)$$

$$\omega = \frac{V_L - V_R}{D} \quad (7)$$

The relation between the vector of velocities and the posture vector  $p$  expressed in the X-Y coordinate has derived as:

$$\dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos(\theta)}{2} & \frac{\cos(\theta)}{2} \\ \frac{\sin(\theta)}{2} & \frac{\sin(\theta)}{2} \\ \frac{1}{D} & -\frac{1}{D} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix} \tag{8}$$

**2.2. The motivation of using left-hand algorithm**

The left-hand algorithm works in such a way that the algorithm checks its left-side and front-side to see if there is a wall with turning to the first non-wall side [19]. If the robot cannot execute any of these movements, it has the option to turn right, as illustrated by the flowchart in Figure 2 [20].

A pseudo-code for the wall follower algorithm with the left-hand rule is as follows:

```
# Wall follower, Left – hand Rule
If left is open Then
turn_left
Else If the front is open Then
go_forward
Else turn_around
Loop
```

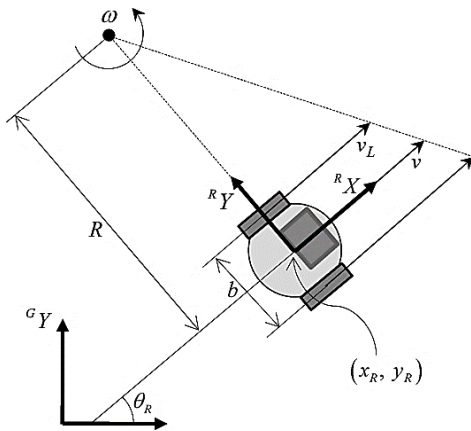


Figure 1. Kinematics of the mobile robot

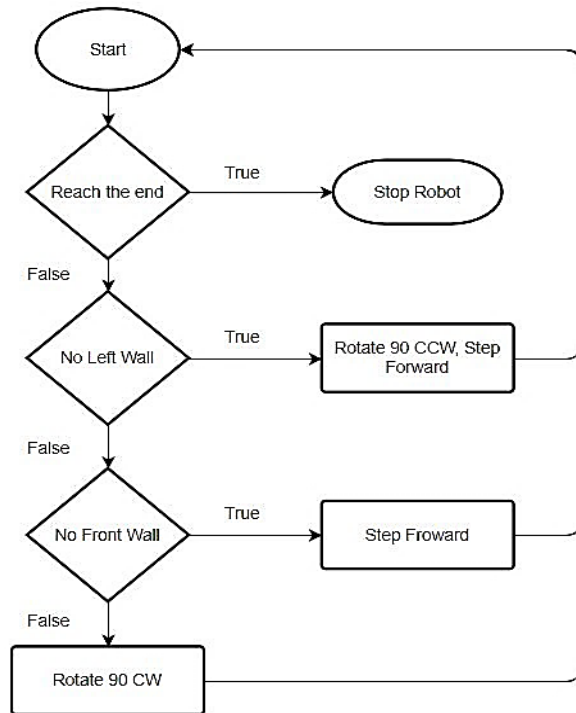
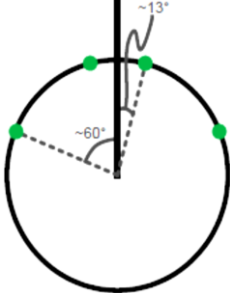
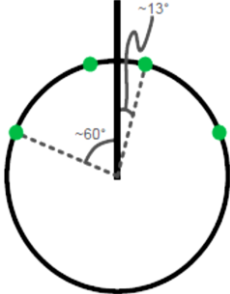
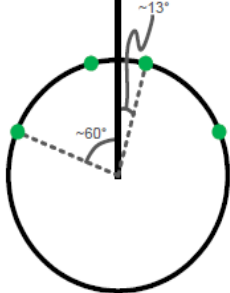


Figure 2. Maze-solving flowchart

**2.3. The motivation of using line following algorithm**

The task of line following is achieved by executing a particular algorithm using two sensors mounted at the bottom of the robot. Pulse-width modulation (PWM) controls the motors' speed and hence, controls the robot's motion and speed of rotation. Three robotic states occur in the following scheme expressed as three output directions: left, right, and forward. Table 1 shows the designing algorithm of line follower control. Figure 3 shows the line follower flowchart. The algorithm gets the data from the two sensors and determines the next motion of the robot.

Table 1. Line follower control algorithm

Statue	Input		Output
	Left sensor	Right sensor	
	0	0	Forward (both motors in the same speed)
	1	0	Increase the speed of right motor
	0	1	Increase the speed of left motor

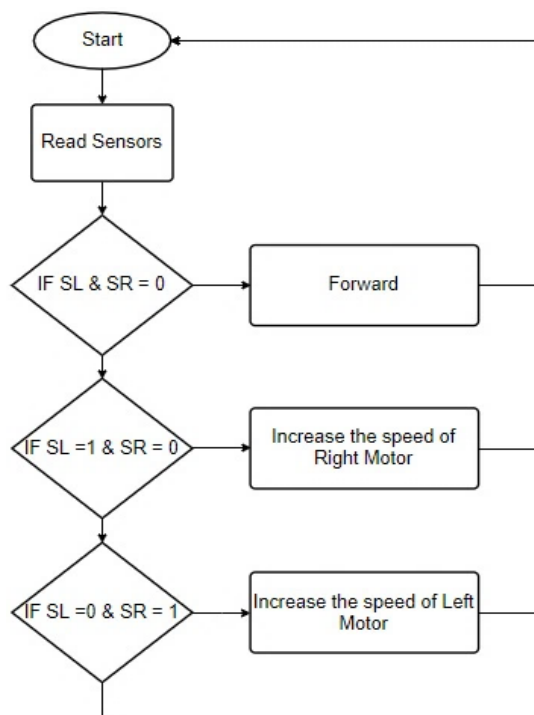


Figure 3. The flowchart of the line following the algorithm

### 3. IROBOT CREATE

The iRobot create is a reprogrammable version of the Roomba vacuum cleaner for researchers, educators, and robotics hobbyists. iRobot create is the most popular design platform used to support research [21], [22] and educational activities in mechatronic and robotic areas [23]-[25]. The iRobot creates components, contain light emitting diode (LEDs), serial port, cliff sensors, speakers, infrared (IR) receiver, and two differentially-driven wheels. The iRobot create can be programmed by serial port directly from the computer through an application programming interface (API) command line. Figure 4 shows the structure of iRobot, Figure 4(a) top-view of the robot and Figure 4(b) bottom-view of the robot platform.

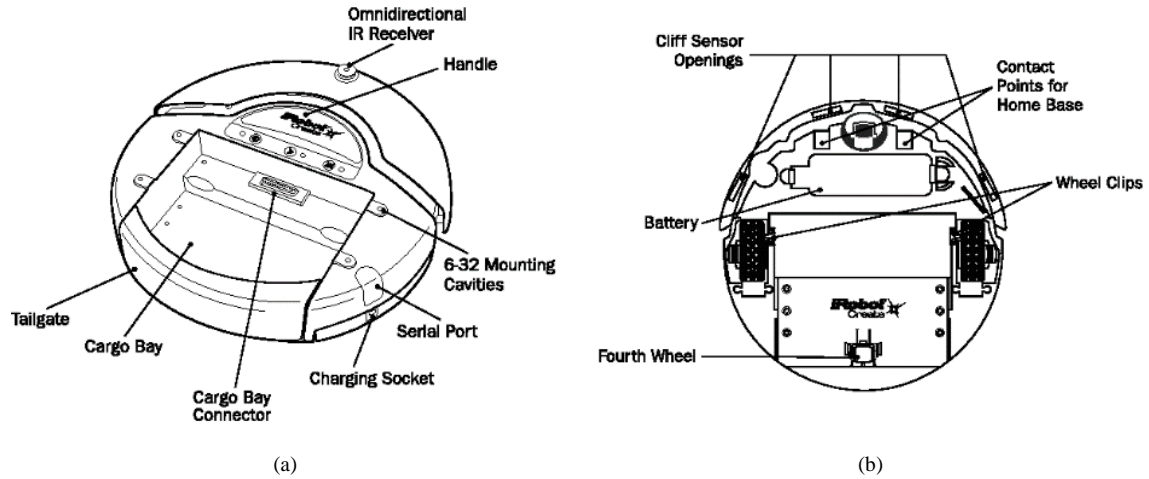


Figure 4. The iRobot structure: (a) top-view of the robot and (b) bottom-view of the robot platform

### 4. MATLAB BASED SIMULATOR FOR IROBOT CREATE

The iRobot create toolbox is a Matlab adapted simulator used for educational purposes. The movements of iRobot create can be simulated and visualized either in autonomous mode or manual mode; the autonomous mode controls the movements of iRobot create by writing a Matlab function program. In contrast, in manual mode, the graphical user interface (GUI) or the keyboard is used to control the movements of the iRobot create [24]. The toolbox contains a main simulator GUI and three GUIs for configuring the settings, replaying the simulation, and creating the map. The Matlab toolbox for the iRobot create (MTIC) toolbox can be used to run the autonomous control program on an actual iRobot [26]. A flowchart of the proposed algorithm is as shown in Figure 5.

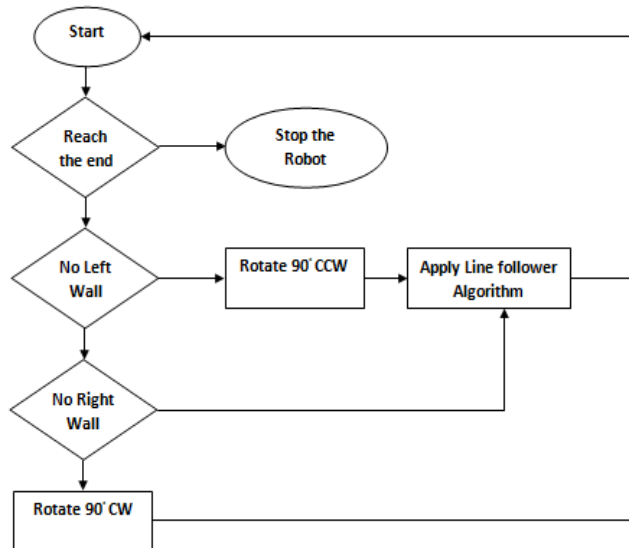


Figure 5. The flowchart of the proposal system

## 5. RESULTS AND DISCUSSION

In this section, the results obtained during this project will be shown. It will start by showing the simulation result for both left-hand wall algorithms. Then it will show the experimental result for the robot on the maze. Figure 6 describes the maze and Figure 7 (see appendix). shows the movement of a robot through the maze. The robot used in the maze has two sonar sensors that start to discover the wall by sending signals and waiting for the reflection, then decide to move depending on the implemented algorithm.

The simplified wall follower algorithm uses the left-hand rule to control the robot's movement through the maze. The robot reads the sensory signals from the left side through the left sensor and then turns left if there is no wall; else, the robot senses at the front to see if the path forward is clear. If the path forward is blocked, the robot will turn right and begin to execute the algorithm from the beginning. The robot also uses the two cliff sensors at the bottom of the robot to discover the line on the white surface, enabling the robot to move in a straight line. The linear velocity of both wheels was set to 0.1 meters per second, and the total time for the robot to complete the mission was  $1.3797e+03$  seconds.

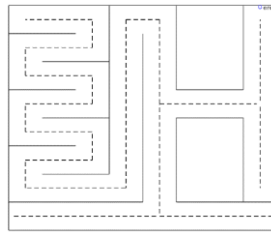


Figure 6. The maze map

## 6. CONCLUSION

The project looks at the famous left-hand maze solving algorithm. The project looks at program design and notes simulation results for essential maze solving steps. The maze robot simulation is realized in this study using a software simulation suite called iRobot create in Matlab. The robot senses the walls before deciding to move in any direction. The robot starts to move in a forward direction. The algorithm begins by checking the left side. If there is a wall, the robot checks the forward side. If the robot senses a wall in front, the robot will turn right, and start to check again. The modified algorithm considers using a line following control algorithm to drive the robot in a straight line; this has two benefits: Firstly, it reduces the number of sensors by one making the robot lightweight and power-efficient. Secondly, it is less costly and less complicated. The simplicity of the program enables the maze robot to explore the maze effectively.

## APPENDIX

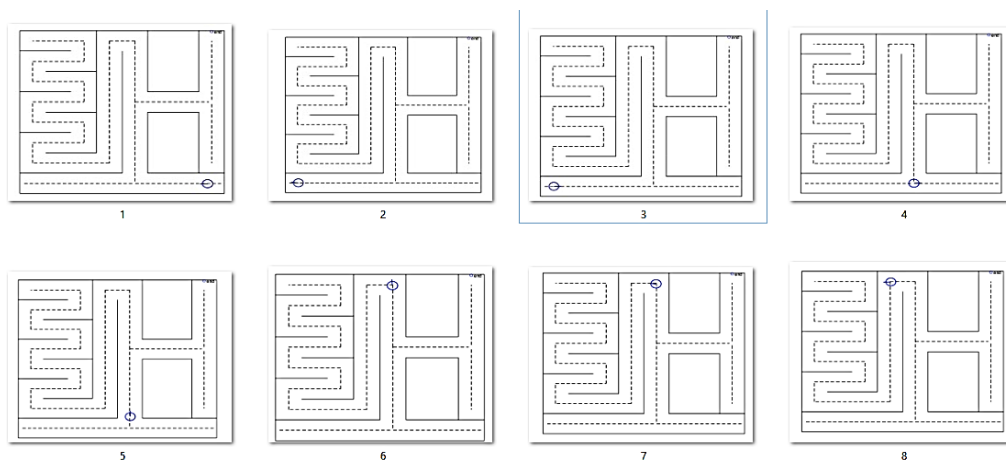


Figure 7. The step of exploring a maze using a proposed algorithm

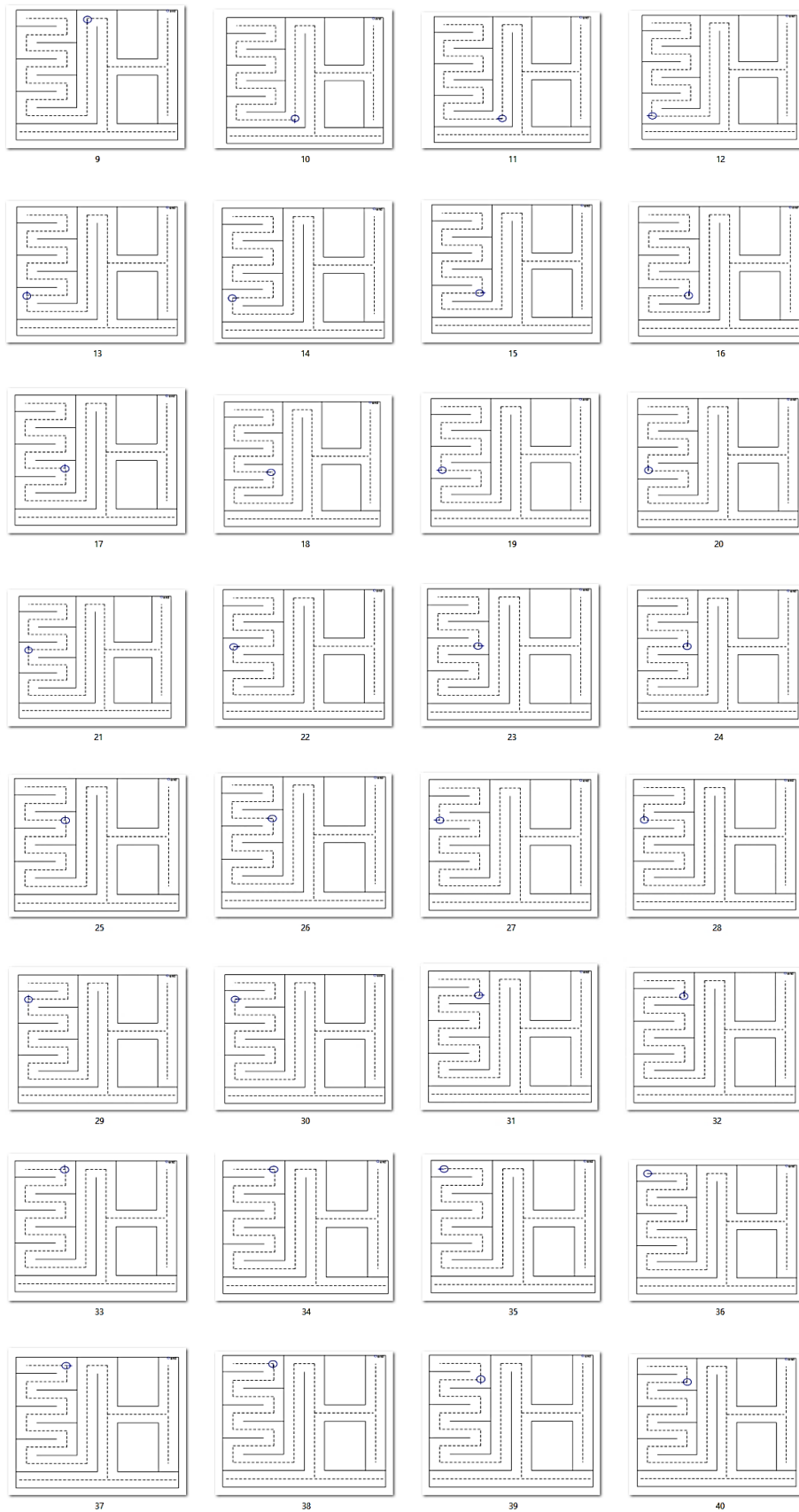


Figure 7. The step of exploring a maze using a proposed algorithm (continue)

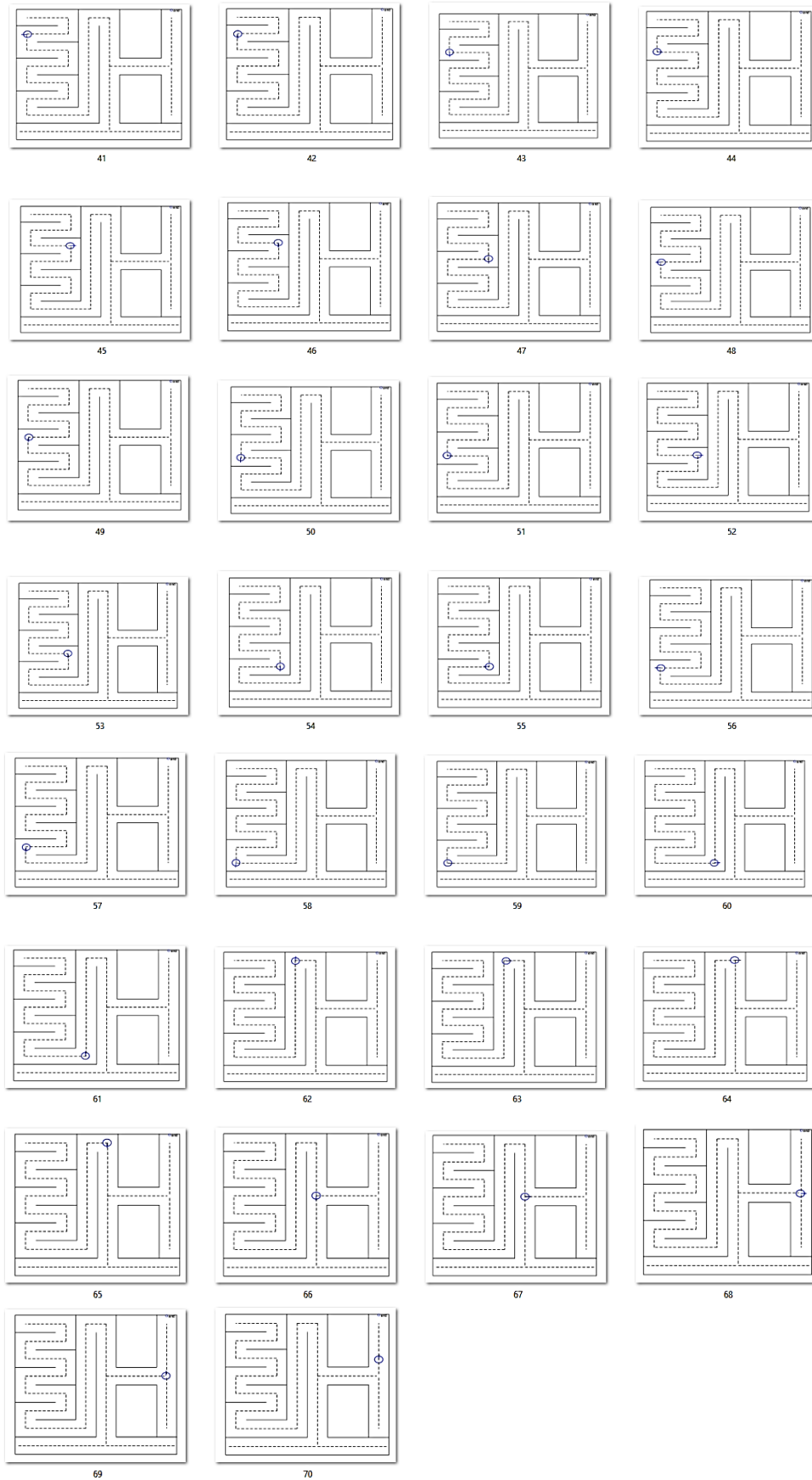





Figure 7. The step of exploring a maze using a proposed algorithm (continue)






## REFERENCES

- [1] N. Seekhao, J. JaJa, L. Mongeau, and N. Y. K. Li-Jessen, "In situ visualization for 3D agent-based vocal fold inflammation and repair simulation," *Supercomputing Frontiers. Innovations*, vol. 4, no. 3, pp. 68–79, 2017, doi: 10.14529/jsfi170304.
- [2] S. Tjiharjadi and E. Setiawan, "Design and Implementation of a Path Finding Robot Using Flood Fill Algorithm," *International Journal of Mechanical Engineering and Robotics Research*, vol. 5, no. 3, pp. 180-185, July 2016, doi: 10.18178/ijmerr.5.3.180-185.
- [3] S. M. J. Jalali, S. Ahmadian, A. Khosravi, S. Mirjalili, M. R. Mahmoudi, and S. Nahavandi, "Neuroevolution-based autonomous robot navigation: A comparative study," *Cognitive System Research*, vol. 62, pp. 35-43, Aug. 2020, doi: 10.1016/j.cogsys.2020.04.001.
- [4] T. Y. Abdalla, A. A. Abed, and A. A. Ahmed, "Mobile robot navigation using PSO-optimized fuzzy artificial potential field with fuzzy control," in *Journal of Intelligent and Fuzzy Systems*, Jan. 2017, vol. 32, no. 6, pp. 3893-3908, doi: 10.3233/IFS-162205.
- [5] M. Ihsan, D. Suhaimi, M. Ramli, S. M. Yuni, and I. Maulidi, "Non-perfect maze generation using Kruskal algorithm," *Jurnal Natural*, vol. 21, no. 1, p. 2020, Feb. 2021, doi: 10.24815/jn.v21i1.18840.
- [6] H. S. Dewang, P. K. Mohanty, and S. Kundu, "A Robust Path Planning for Mobile Robot Using Smart Particle Swarm Optimization," in *Procedia Computer Science*, Jan. 2018, vol. 133, pp. 290-297, doi: 10.1016/j.procs.2018.07.036.
- [7] D. C. Dracopoulos, "Robot path planning for maze navigation," *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, 1998, pp. 2081-2085 vol.3, doi: 10.1109/IJCNN.1998.687180.
- [8] M. O. A. Aqel, A. Issa, M. Khair, M. ElHabbash, M. AbuBaker, and M. Massoud, "Intelligent Maze Solving Robot Based on Image Processing and Graph Theory Algorithms," *2017 International Conference on Promising Electronic Technologies (ICPET)*, 2017, pp. 48-53, doi: 10.1109/ICPET.2017.15.
- [9] M. M. Rehaan, R. A. Altahtawy, A. R. Ibrahim, C. M. Elias, D. M. Mahfouz, and E. I. Morgan, "Studying the Dynamical-based Closed Loop Robot Trajectory Behavior via LQR, SMC, and TDC," *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2019, pp. 1-6, doi: 10.1109/ICVES.2019.8906380.
- [10] A. A. Dahalan, A. Saudi, and J. Sulaiman, "Robot pathfinding with obstacle avoidance capabilities in a static indoor environment via TOR iterative method using harmonic potentials," *The 16th IMT-GT International Conference on Mathematics, Statistics and their Applications (ICMSA 2020) ITM Web Conferences*, Jan. 2021, vol. 36, pp. 04006, doi: 10.1051/itmconf/20213604006.
- [11] K. Vennela, M. C. Chinnaiah, S. Dubey, and S. Savithri, "Implementation of Mobile Robot Navigation Mechanism Using FPGA: An Edge Detection-Based Approach," in *International Conference on Innovative Computing and Communications*, 2019, vol. 55, pp. 215–222, doi: 10.1007/978-981-13-2324-9\_21.
- [12] R. H. Abiyev, M. Arslan, I. Gunsel, and A. Cagman, "Robot Pathfinding Using Vision Based Obstacle Detection," *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, 2017, pp. 1-6, doi: 10.1109/CYBCONF.2017.7985805.
- [13] R. Kumar *et al.*, "Maze solving robot with automated obstacle avoidance," *Procedia Computer Science*, vol. 105, pp. 57-61, 2017, doi: 10.1016/j.procs.2017.01.192.
- [14] S. Tjiharjadi, "Design and Implementation of Flood Fill and Pledge Algorithm for Maze Robot," *International Journal of Mechanical Engineering and Robotics Research*, vol. 8, no. 4, 2019, doi: 10.18178/ijmerr.8.4.632-638.
- [15] T. A. Salh and M. Z. Nayef, "Intelligent surveillance robot," *2013 International Conference on Electrical Communication, Computer, Power, and Control Engineering (ICECCPCE)*, 2013, pp. 113-118, doi: 10.1109/ICECCPCE.2013.6998745.
- [16] A. Srebro, "Fault-tolerant algorithm for a mobile robot solving a maze," *Challenges of Modern Technology*, vol. 4, no. 1, pp. 21-29, 2013. [Online]. Available: <https://bibliotekanauki.pl/articles/115441>.
- [17] R. R. Serrezuela, A. F. C. Chávarro, M. A. T. Cardozo, A. L. Toquica, and L. F. O. Martinez, "Kinematic modelling of a robotic ARM manipulator using MATLAB," *ARN Journal of Engineering and Applied Sciences*, vol. 12, no. 7, 2017, Accessed: April 27, 2021. [Online]. Available: [http://www.arnjournals.org/jeas/research\\_papers/rp\\_2017/jeas\\_0417\\_5860.pdf](http://www.arnjournals.org/jeas/research_papers/rp_2017/jeas_0417_5860.pdf)
- [18] M. Cui, H. Liu, W. Liu, and Y. Qin, "An Adaptive Unscented Kalman Filter-based Controller for Simultaneous Obstacle Avoidance and Tracking of Wheeled Mobile Robots with Unknown Slipping Parameters," *Journal of Intelligent & Robotic Systems*, vol. 92, no. 3–4, pp. 489–504, Dec. 2018, doi: 10.1007/s10846-017-0761-9.
- [19] A. I. Rusu, A. Florescu and S. G. Rosu, "Efficient Navigation of a Robot Based on an Improved Contrast of Colours Algorithm," *2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 2018, pp. 37-42, doi: 10.1109/SIITME.2018.8599202.
- [20] A. Zarkasi, H. Ubaya, C. D. Amanda, and R. Firsandaya, "Implementation of ram based neural networks on maze mapping algorithms for wall follower robot," in *Journal of Physics: Conference Series*, vol. 1196, no. 1, Apr. 2019, doi: 10.1088/1742-6596/1196/1/012043.
- [21] R. C. Hsu, P. -C. Su, J.-L. Hsu and C.-Y. Wang, "Real-Time Interaction System of Human-Robot with Hand Gestures," *2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, 2020, pp. 396-398, doi: 10.1109/ECICE50847.2020.9301957.
- [22] V. Popov, N. Shakev, S. Ahmed, and A. Toplaov, "Recognition of Dynamic Targets using a Deep Convolutional Neural Network," *ANNA '18: Advances in Neural Networks and Applications 2018*, 2018, pp. 1-6.
- [23] O. A. Ejofodomi and G. Ofualagba, "Exploring the feasibility of robotic pipeline surveillance for detecting crude oil spills in the Niger delta," *International Journal of Unmanned Surveillance Systems Engineering (IJUSEng)*, vol. 5, no. 3, pp. 38-52, 2017, doi: 10.14323/ijuseng.2017.7.
- [24] J. E. Inglett and E. J. Rodríguez-Seda, "Object transportation by cooperative robots," *SoutheastCon 2017*, 2017, pp. 1-6, doi: 10.1109/SECON.2017.7925348.
- [25] N. Aphiratsakun and S. Liwsakphaiboon, "Path Control iRobot Create2 Based Sensors," *Proceedings of Engineering and Technology Innovation*, Aug. 2020, vol. 16, pp. 45–51, doi: 10.46604/peti.2020.4130.
- [26] S.-K. Su, "Model-Based Development of Multi-iRobot Simulation and Control," *Arizona State University*, 2012. [Online]. Available: [https://keep.lib.asu.edu/\\_flysystem/fedora/c7/66132/tmp/package-PBGTOa/Su\\_asu\\_0010N\\_12323.pdf](https://keep.lib.asu.edu/_flysystem/fedora/c7/66132/tmp/package-PBGTOa/Su_asu_0010N_12323.pdf)




**BIOGRAPHIES OF AUTHORS**

**Ibrahim Majid Mohammed**    received the B.Eng. degree in computer and communication engineering from Al-Rafidain University College, Baghdad, Iraq, in 2016, and the master's degree in computer and communication engineering from Universiti Malaysia Perlis, Perlis, Malaysia, in 2020. He is currently pursuing the Ph.D. degree in electrical and electronic engineering with Universiti Sains Malaysia (USM), Penang, Malaysia. His research interests include image processing and intelligent systems. He can be contacted at email: [ibrahim.m.94@student.usm.my](mailto:ibrahim.m.94@student.usm.my).






**Mustafa Zuhaer Nayef Al-Dabagh**    (Member, IEEE) received the bachelor and master's degree in computer engineering technology from Northern Technical University (NTU), Iraq, in 2008 and 2014. His research interests include Computer vision, image processing and machine learning. He can be contacted at email: [mustafa.zuhaer.nayef@gmail.com](mailto:mustafa.zuhaer.nayef@gmail.com).



**Salar Jamal Rashid**    received the B.Sc. and M.Sc. degree in Computer Engineering Technology from Northern Technical University, Iraq in 2011 and 2016 respectively. His current research interests include Internet of Things, Computer Networks, Pattern Recognition, Information, and Communication Technology. He can be contacted at email: [salar.jamal@ntu.edu.iq](mailto:salar.jamal@ntu.edu.iq).



**Nor Ashidi Mat Isa**    (Member, IEEE) received the B.Eng. degree (Hons.) in electrical and electronic engineering from Universiti Sains Malaysia (USM), in 1999, and the Ph.D. degree in electronic engineering (majoring in image processing and artificial neural network), in 2003. He is currently a Professor and the Deputy Dean (Academic, Career, and International) with the School of Electrical and Electronic Engineering, USM. His research interests include intelligent systems, image processing, neural networks, biomedical engineering, intelligent diagnostic systems, and algorithms. He can be contacted at email: [ashidi@usm.my](mailto:ashidi@usm.my).