Journal of Robotics and Control (JRC)
Volume 3, Issue 1, January 2022
ISSN: 2715-5072 DOI: 10.18196/jrc.v3i1.12241

A Simulation-Based Study of Maze-Solving-Robot Navigation for Educational Purposes

Ismu Rijal Fahmi [1], Dwi Joko Suroso [2*]
[1,2] Dept. of Nuclear Engineering and Engineering Physics, Universitas Gadjah Mada, Yogyakarta, Indonesia
Email: [1]ismu.rijal.fahmi@mail.ugm.ac.id, [2]dwi.jokosuroso@ugm.ac.id
[*]Corresponding Author

***Abstract*—The point of education in the early stage of studying robotics is understanding its basic principles joyfully. Therefore, this paper creates a simulation program of indoor navigations using an open-source code in Python to make navigation and control algorithms easier and more attractive to understand and develop. We propose the maze-solving-robot simulation as a teaching medium in class to help students imagine and connect the robot theory to its actual movement. The simulation code is built for free to learn, improve, and extend in robotics courses or assignments. A maze-solving robot study case is then done as an example of implementing navigation algorithms. Five algorithms are compared, such as Random Mouse, Wall Follower, Pledge, Tremaux, and Dead-End Filling. Each algorithm is simulated a hundred times in every type of the proposed mazes, namely mazes with dead ends, loops only, and both dead ends and loops. The observed indicators of the algorithms are the success rate of the robots reaching the finish lines and the number of steps taken. The simulation results show that each algorithm has different characteristics that should be considered before being chosen. The recommendation of when-to-use the algorithms is discussed in this paper as an example of the output simulation analysis for studying robotics.**

Keywords—Robot simulation; Maze-solving robot; Random mouse; Wall follower; Pledge; Tremaux; Dead-end filling

I. INTRODUCTION

Robots are all forms of machines that automatically replace human effort, including those whose physic or function does not always resemble humans [1]. Some essential robot abilities are interacting with the surrounding environment and moving, conforming with it. Therefore, the robots must be equipped with navigation ability. Navigation is the process of directing a robot to pass through a specific environment [2]. The environment can be simple, like a single track, or complicated, such as a maze [3].

When the robot interacts with the environment, it needs to be equipped with an embedded algorithm in the source code. The source code contains a series of actions or reactions to an event, which can be simple logic, such as finite state machines or a series of complex systems [4].

The robot navigation needs to know the surrounding, which can be partial or a whole and has a series of control systems equipped to support [5], [6]. Some advantage automation for navigation can be used the wireless-based sensor to communicate or using some advanced vision sensor and algorithms [7], [8]. The author's previous work also emphasized indoor localization for a small object, i.e., a mobile robot based on the received signal from a wireless device [9], [10], and comparisons of certain positioning algorihm for distance or range-based [11]. We also constructed the simple line-following robot to test our algorithm for both stationary and moving robot tracking [12].

This paper discusses the maze-solving-robot algorithms by simulation. Unlike previous standards in robotics simulations, i.e., [13], [14], we are eager to explain the algorithms in visualization to students with a straightforward method. Especially students who are new to robotics can absorb the material efficiently [15]. We aim that first, the students might be interested and pay more attention to the learning process. This paper is made to demonstrate how to study and understand the robot's motion planning and navigation with a simple and interactive maze-solving simulation. Based on our concerns, a limited study shows some algorithms applied using free and straightforward software, i.e., Python [16]–[18]. With the free and open software, the funding and difficulty barriers of learning robotics at the first concern can be removed. As a learning model using this program, maze-solving algorithms are compared. With this study case, the characteristic of each algorithm is identified when it interacts with various mazes. This identification will help students implement a suitable algorithm for the maze type.

It is also the case that novice programming can learn faster by simple robotics, as proven in [19]. Simulating robots is one of the good examples that learners can use in two ways: first, study programming, and second, understand the essential part of robotics. By studying the robotics simulation to solve the maze, the learning process can be scaled up to build a straightforward robot, i.e., line-following robots. In education and robot learning, the sparks of interest in the novice learner can significantly impact the subsequent robotics learning. In the first stage, the simulation can also reduce the risk of a bug or error in the robot before building the real one [20].

This paper considered five algorithms used commonly in maze robot navigation. The algorithms are Random Mouse Algorithm (RMA), Wall Follower Algorithm (WFA) [21],

[22], Pledge Algorithm (PA), Tremaux Algorithm (TA), and Dead-End Filling Algorithm (DEFA). Meanwhile, the mazes are varied based on the finish point location and whether they have dead ends, loops, or both. The maze type variation will give insight into how a robot with a particular algorithm interacts with obstruction in the maze. Another algorithm, i.e., flood filling, is similar to our variation in the maze for dead ends and loop [23], [24].

The paper is structured as follows. We introduce the paper topic in the first part of the paper. In the second part, we discussed the algorithms applied for maze-solving-robot simulation. The next part discusses the maze structures and variations. In the fourth part, we detail the simulation scenario and the implemented algorithm flowcharts. The fifth part discusses the simulation results, and lastly, we conclude the finding in the conclusion section.

II. ALGORITHMS

This paper compares the performance of the algorithms of RMA, WFA (Left and Right combined), PA (Left and Right combined), TA, and DEFA that follow the basic concept of Finite State Machine (FSM). A general example of FSM can be represented by state(s) and transition(s). Fig. 1 shows an instance of FSM with three states, namely S_1, S_2 and S_3 and four transitions, specifically T_{1a}, T_{1b}, T_2 and T_3 [25]. The graph nodes or circles represent the states, while the graph edges or arrows represent the transitions. For example, if there is a "start" trigger in S_1, the robot state will shift from S_1 to S_2 and S_3. Those shifting processes are called transition [25], [26].

FSM is a form of automata in robotics. It is powerful yet relatively simpler to be applied than other automata types, i.e., push down and turning machine. For robotics, FSM is sufficient to help complete a repetitive task. In this regard, FSM can help understand robot behavior as this paper's comparative study objective.

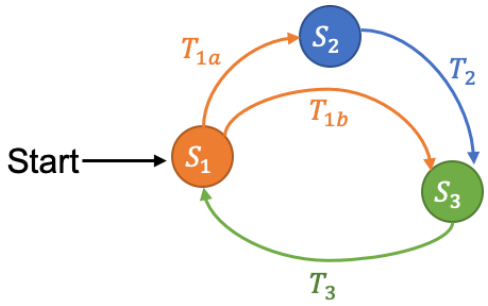

Fig. 1. FSM diagram example [25].

A. Random Mouse Algorithm (RMA)

RMA is an algorithm that moves robots without any specific guides. With this algorithm, the robots will go in a random direction whenever they meet an intersection. Without intersections, the robots will follow the path and never return unless they meet a dead end [4]. With its characteristics, this algorithm is considered the most unintelligent.

B. Wall Follower Algorithm (WFA)

WFA is an algorithm that moves robots using a wall as a guide. There are two types of this algorithm: Left WFA and Right WFA [3]. Like its name, the Left WFA makes the robots follow the left wall, and the right one makes the robots follow the right wall. With this algorithm, the robot may not find a way out if the followed wall is not connected to others.

C. Pledge Algorithm (PA)

Pledge Algorithm (PA) is an improved algorithm of WFA [27]. PA has two types as well, i.e., Left PA and Right PA. PA makes the robots follow the wall like WFA, but PA has a mechanism for the robots to exit a looping wall. It is possible by counting a number representing the turns. Initially, the value of this number is 0. This number will be added or subtracted by 1 (one) if the robots make a turn. For example, turning left is associated with subtraction, and turning right is associated with addition. This correspondence may be interchanged as long as it is consistent throughout the maze-solving process. This rule will automatically return the number to 0 when the robots face the original direction. When it happens, PA tells the robots to exit the followed wall by turning in the opposite direction at the next intersection. If the robot passes a turn but not an intersection, the PA algorithm then commands the to turn around and follow a wall across the previous one.

D. Tremaux Algorithm (TA)

The basic rule of Tremaux Algorithm (TA) is that robots cannot pass through a path more than twice. Therefore, TA requires robots to record their paths in memory [28]. The paths can be distinguished into three types [27]:

- Unmarked, meaning that a path is not explored yet.
- It was marked once, meaning that a path was passed once.
- It was marked twice, meaning that a path was passed twice.

If a path was marked twice, that path is considered wrong and will not be visited anymore. This marking procedure makes unnecessary multiple visits to the same path eliminated. At the end of exploration, TA leaves a continuous once-passed mark on a path connecting the starting and the finish point [27].

E. Dead-End Filling Algorithm (DEFA)

Dead-End Filling Algorithm (DEFA) is an algorithm requiring robots to know the maze before starting to explore. It can be done by equipping the robots with scanners above the maze that detect every dead end. If a dead-end is found, the scanner will mark and track it to the nearest intersection [29]. That path is then marked and will later not be visited by the robots. An intersection can be marked not to be visited as well if all the paths joining in the intersection lead to dead ends. Nevertheless, DEFA may leave an intersection with two or more unmarked paths, especially when the paths contain loops. In that case, DEFA leaves the robots unguided and makes them choose a path randomly.

III. MAZE

IV. The mazes used in this paper are in the size of 81×61 and distinguished into six categories. These categories are generated based on the combination of the finish position and

whether the maze has dead ends, loops, or both. The first finish position is located in the maze's bottom-right corner, where the robots are said to reach finish when they exit the maze. Meanwhile, the finish location of the second type is in the center of the maze. The starting points for the blue-colored robots are the same in all the mazes, as shown in Fig. 2 to Fig. 7.

A. *Bottom-Right Corner (BRC) Exit*

The maze exit position is located in the bottom-right corner of each maze, as shown in Fig. 2 to Fig. 4.

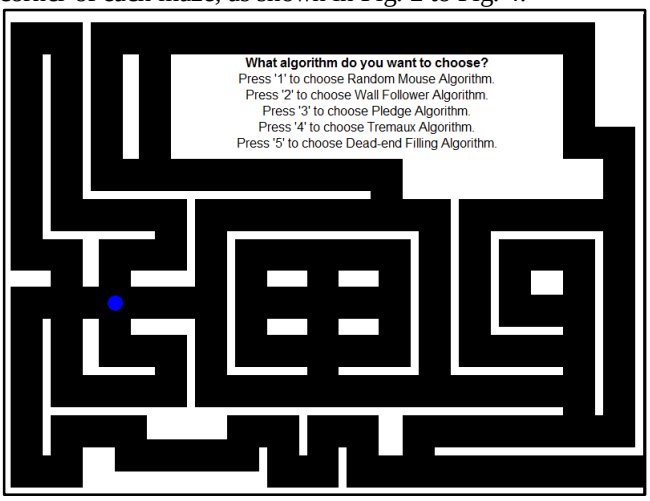

Fig. 2. Maze with loops only and an exit way in the bottom-right corner

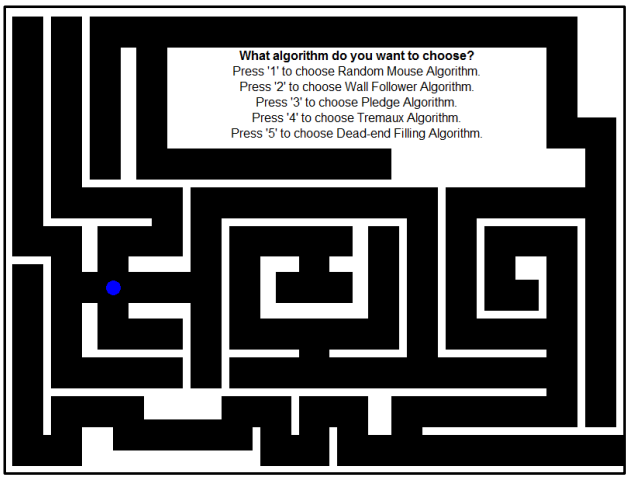

Fig. 3. Maze with dead ends only and an exit way in the bottom-right corner

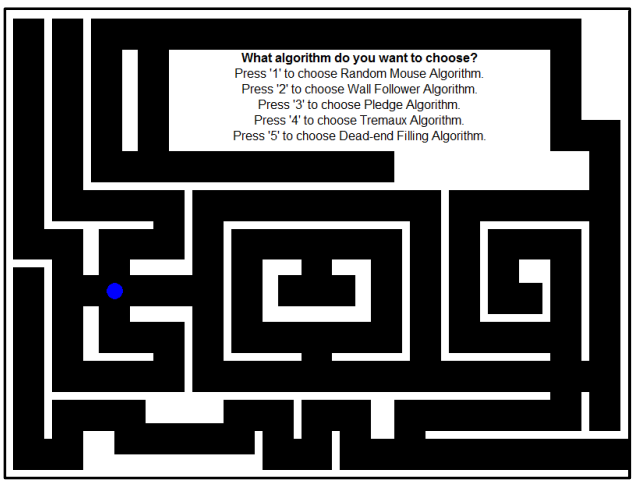

Fig. 4. Maze with an exit way in the bottom-right corner and a mix of loops and dead ends

B. *Center Finish Point (CFP)*

The algorithm needs to find the finish point in the center of the maze, as shown in Fig. 5 to Fig. 7.

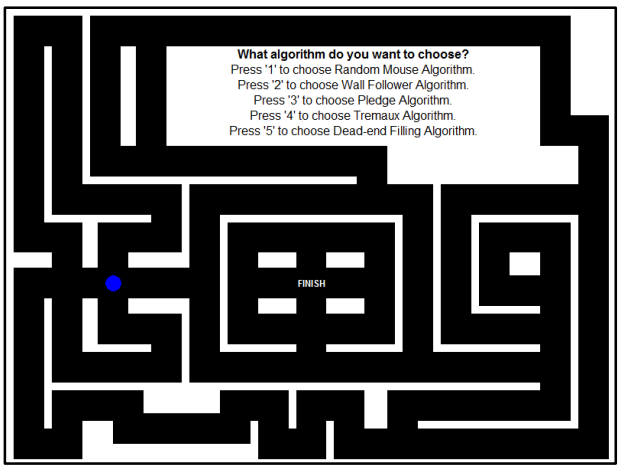

Fig. 5. Maze with loops only and a finish point in the center

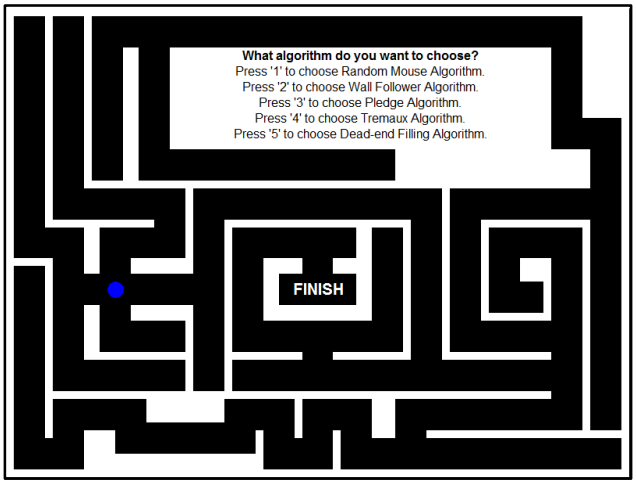

Fig. 6. Maze with dead ends only and a finish point in the center

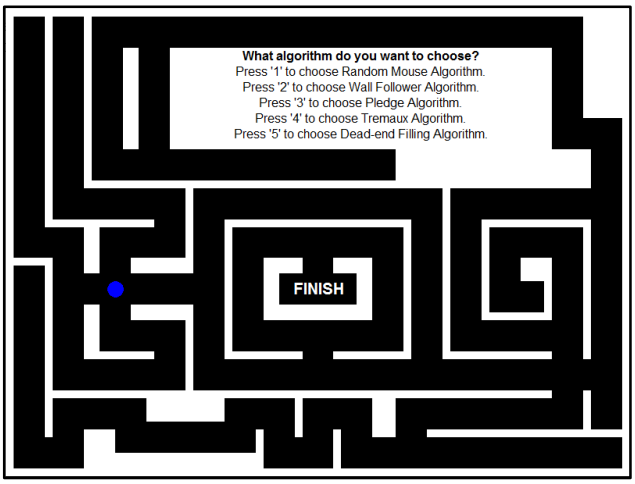

Fig. 7. Maze with a finish point in the center and a mix of loops and dead ends

V. PROGRAM SIMULATION

This paper used Python with the library of graphics.py to develop the simulation program. The way of the algorithms work was translated to the flow charts shown in Fig. 8 - Fig. 12 and run a hundred times for each maze. The flowchart of the right-wall following algorithm is the same as Fig. 9, except the left wall is replaced by the right wall. Meanwhile, the Right Pledge Algorithm has a similar flowchart, with the word "left wall" is replaced with "right wall" and "Number=Number+2" with "Number=Number-2".

In every simulation, the program notes whether the robots succeed in reaching the finish or not and the number of steps taken. The success rate shown at the end of the simulations represents the algorithm's reliability, while the taken-step numbers show how efficient the algorithm is.

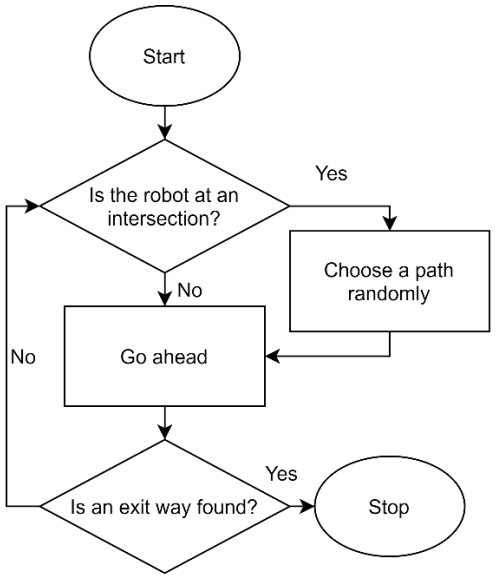

Fig. 8. RMA flowchart

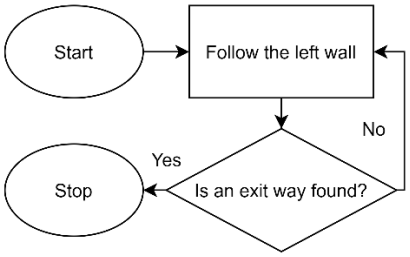

Fig. 9. Left WFA flowchart

VI. RESULTS AND DISCUSSIONS

The simulation results are shown in Fig. 13 and Fig. 14. Fig. 13 and Fig. 14 show that RMA works for any maze, but it takes a long time to solve. It always took the most steps to solve the maze except slightly fewer than DEFA in the BRC Loop maze. Besides, DEFA is lower in the BRC with loop only because it runs as disorderedly as RMA since it cannot detect and block any dead-end in that maze. Moreover, RMA performed the worst in the CFP Loop because it took the wrong intersection most of the time and went to the edge aisles with relatively long distances. This simulation implied that RMA is suitable for a simple maze-solving robot project where speed is not essential.

Fig. 13 shows that WFA is the least reliable algorithm because it will not work when the start and finish points are not connected with a continuous wall, such as CFP Mix and Loops. On the other hand, the success rate of WFA in BRC Mix is close to a quarter because there is a wall out of four that can guide the robot to the finish point. Similarly, BRC Loop has a success rate of 70% since three out of the four adjacent walls in the starting point connect with the finishing hole. However, once the WFA works, it can swiftly lead the robot, as shown in Fig. 14, since the robot will not wander the maze, similar to other algorithms.

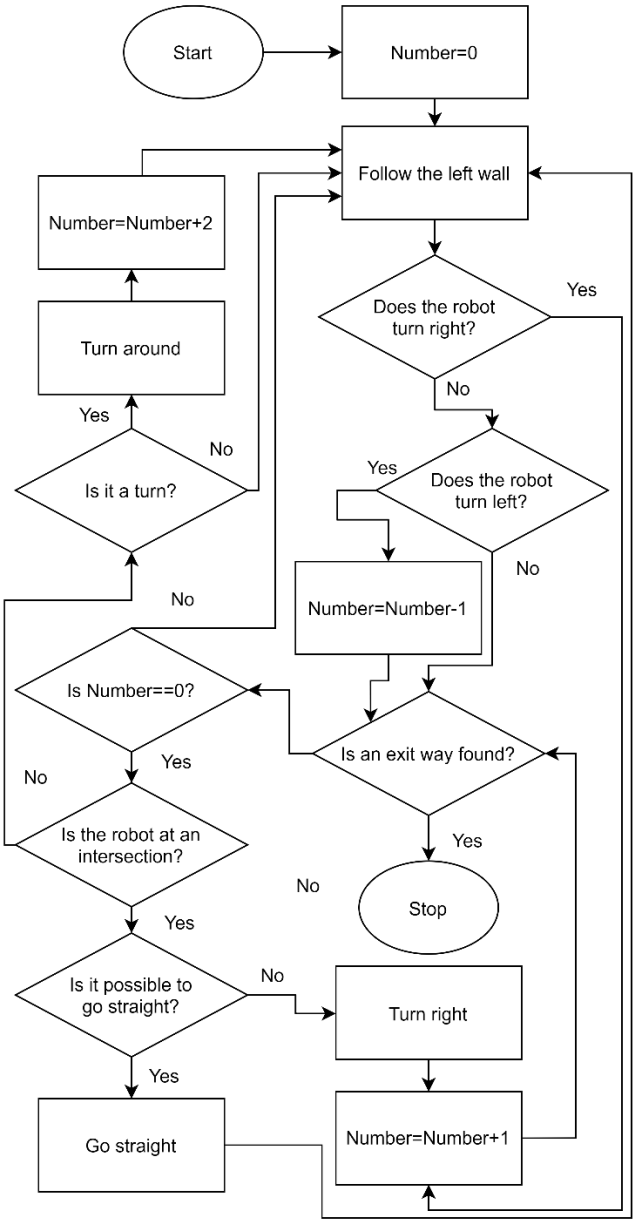

Fig. 10. Left PA flowchart

As an improved version of WFA, PA has perfect success rates in the BRC Mix and Loop mazes, better than WFA. Nevertheless, it cannot improve the robot performances in the CFP Mix and CFP Loop as it cannot guide the robot back to the internal loop once the robot reaches the outer wall. Regarding the required steps to reach the finish, Fig. 14 illustrates that PFA has similar numbers to WFA except for the CFP Dead End. PA recognizes repetitive turns and exits

the followed wall that makes the robot go through a longer path.

Fig. 13 shows that TA effectively solves any mazes as it always has a 100% success rate. The results show that the speed is not affected by the maze type, as illustrated in Fig. 14. The TA algorithm focuses on the paths instead of the walls. As long as the mazes have the same size, the robot with TA needs roughly the same steps to reach the finish. Nevertheless, TA tends to explore the mazes more broadly than WFA and PA because of the random turns it takes when there is an intersection with more than one unpassed path.

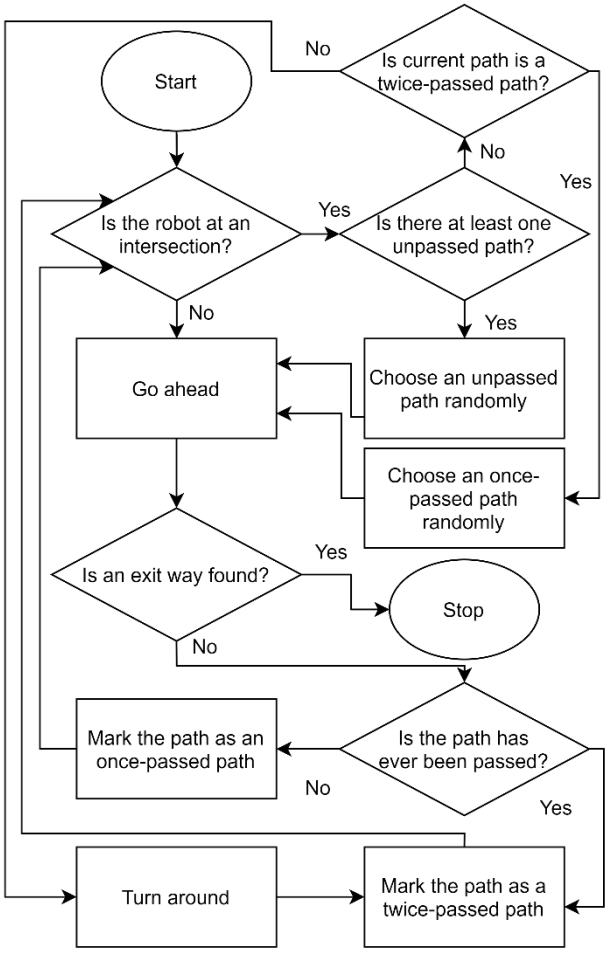

Fig. 11. TA flowchart

According to Fig. 13, DEFA is a reliable algorithm in the same way as RMA and TA. However, DEFA is efficient only in the maze with dead ends. In other words, the more loops the maze has, the more DEFA's performance is like RMA's as it turns to a random direction when there is more than an unblocked path in an intersection. That phenomenon is seen in Fig. 14, where DEFA took the fewest steps in the CFP and BRC Dead-End while it did the most in BRC Loop. To compare our results to a recently published paper related to maze-solving robot simulation, we have a more relatively high number of data in terms of the number of steps, and we do not consider the timing process like in [30].

The discussion above is an example of the simulation-output-data analysis that can help teachers illustrate each algorithm's characteristics or students learn robotics by themselves. As this program was developed using an open-source and simple code, they may modify and develop it according to their needs. In summary, the simulation results are shown in Table I and Table II.

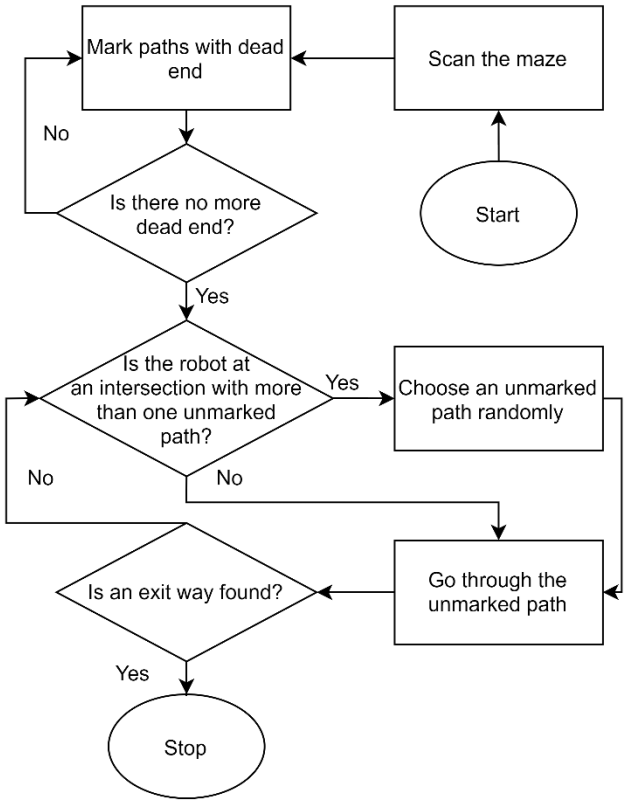

Fig. 12. DEFA flowchart

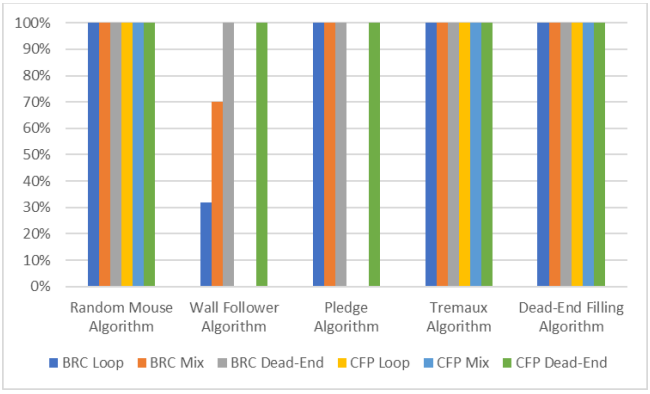

Fig. 13. Success rates of each algorithm in each maze

VII. Conclusion

This paper creates simulation-based indoor navigations that can introduce robotics to students more easily and enjoyably. A comparative study of several navigation algorithms is then done using this program to know when the algorithms should be applied. The study case clearly shows that the algorithm should be chosen based on the type of maze explored. RMA will work in any mazes, but speed does not matter, and simplicity is highly considered. WFA will guide the robots rapidly to the finish if a continuous wall connects the start to the finish point. PA will be suitable if speed is essential, but the robots will encounter separated walls to reach the edge of the maze. TA is a good choice for the robots that will perform in various mazes with moderate speed, especially for mazes with many loops. Moreover, TA is a safe

choice when the characteristic of the maze cannot be identified well before being explored as it is barely affected by the maze type. Lastly, DEFA is a good option if the mazes consist primarily of dead ends.

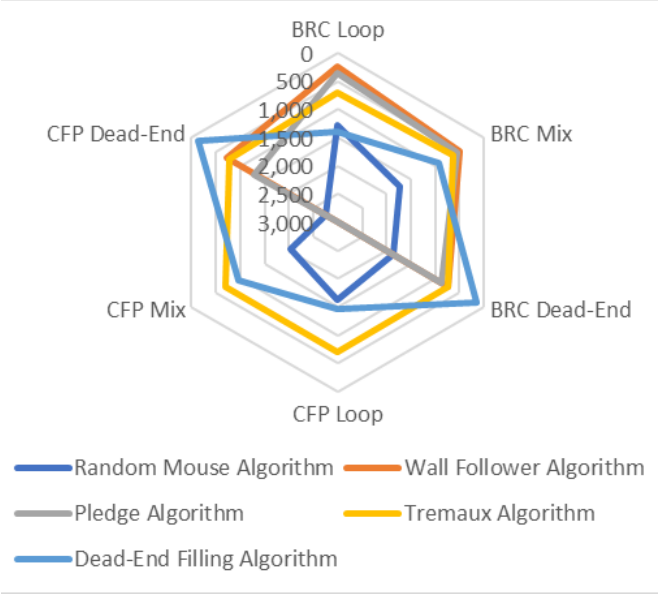

Fig. 14. The number of taken steps of each algorithm in each maze

TABLE I. RESULTS FOR BOTTOM-RIGHT CORNER EXIT

Maze	Parameter	RMA	WFA	PA	TA	DEFA
Loop	No. of taken steps	1284.6	241.8	367.2	712.02	1388.3
	No. of passed intersections	49.61	5.50	11.44	25.12	54.01
	Success rate	100%	32%	100%	100%	100%
Mix	No. of taken steps	1727.4	504.9	542.8	626.3	918.5
	No. of passed intersections	55.42	13.76	15.50	18.55	35.12
	Success rate	100%	70%	100%	100%	100%
Dead-End	No. of taken steps	1870.7	730.1	867.6	733.1	153
	No. of passed intersections	50.55	20.52	23.92	18.37	5
	Success rate	100%	100%	100%	100%	100%

TABLE II. RESULTS FOR CENTER FINISH POINT

Maze	Parameter	RMA	WFA	PA	TA	DEFA
Loop	No. of taken steps	1626	-	-	710	1460.8
	No. of passed intersections	47.31	-	-	22.11	42.76
	Success rate	100%	0%	0%	100%	100%
Mix	No. of taken steps	2035.7	-	-	711.9	960.4
	No. of passed intersections	52.72	-	-	19.93	37.14
	Success rate	100%	0%	0%	100%	100%
Dead-End	No. of taken steps	2758.7	729.7	1296.8	773.5	130
	No. of passed intersections	65.81	19.59	34.73	18.98	6.00
	Success rate	100%	100%	100%	100%	100%

REFERENCES

[1] J. C. Giger, N. Piçarra, P. Alves-Oliveira, R. Oliveira, and P. Arriaga, "Humanization of robots: Is it really such a good idea?," *Human Behavior and Emerging Technologies*, vol. 1, no. 2. pp. 111–123, 2019.

[2] F. Gul, W. Rahiman, and S. S. Nazli Alhady, "A comprehensive study for robot navigation techniques," *Cogent Engineering*, vol. 6, no. 1. 2019.

[3] A. M. J. Sadik, M. A. Dhali, H. M. A. B. Farid, T. U. Rashid, and A. Syeed, "A comprehensive and comparative study of maze-solving techniques by implementing graph theory," *Proc. - Int. Conf. Artif. Intell. Comput. Intell. AICI 2010*, vol. 1, no. November, pp. 52–56, 2010.

[4] A. Halma, E. Bovenkamp, and J. van Oorschot, "RoboMind Challenges Maze Solving," *RoboMind Academy*, 2012. .

[5] Si. F. R. Alves, J. M. Rosario, Hu. F. Filho, L. K. . RIncon, and R. A. . Yamasaki, "Conceptual Bases of Robot Navigation Modeling, Control and Applications," *Adv. Robot Navig.*, vol. 1, no. June, pp. 3–28, 2011.

[6] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, pp. 1–22, 2019.

[7] J. Biswas and M. Veloso, "WiFi localization and navigation for autonomous indoor mobile robots," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 4379–4384, 2010.

[8] F. Fraundorfer *et al.*, "Vision-based autonomous mapping and exploration using a quadrotor MAV," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4557–4564, 2012.

[9] D. J. Suroso, P. Cherntanomwong, P. Sooraksa, and J. Takada, "Location fingerprint technique using Fuzzy C-Means clustering algorithm for indoor localization," in *TENCON 2011 - 2011 IEEE Region 10 Conference*, 2011, pp. 88–92.

[10] D. J. Suroso, M. Arifin, and P. Cherntanomwong, "Distance-based Indoor Localization using Empirical Path Loss Model and RSSI in Wireless Sensor Networks," *J. Robot. Control*, vol. 1, no. 6, pp. 199–207, 2020.

[11] F. Y. M. Adiyatma, A. E. Kurniawan, D. J. Suroso, and P. Cherntanomwong, "Performance Comparison of Several Range- based Techniques for Indoor Localization Based on Received Signal Strength Indicator," vol. 7, no. 1, pp. 40–53, 2021.

[12] P. Cherntanomwong and D. J. Suroso, "Indoor localization system using wireless sensor networks for stationary and moving target," *2011 8th Int. Conf. Information, Commun. Signal Process.*, no. 1, pp. 1–5, 2011.

[13] O. Michel, "WebotsTM: Professional Mobile Robot Simulation," vol. 1, no. 1, pp. 39–42, 2004.

[14] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 1321–1326, 2013.

[15] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, no. i, pp. 51416–51431, 2021.

[16] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011.

[17] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 99–104, 2007.

[18] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[19] L. Major, T. Kyriacou, and O. P. Brereton, "Systematic literature review: Teaching novices programming using robots," *IET Semin. Dig.*, vol. 2011, no. 1, pp. 21–30, 2011.

[20] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand, "Can robot navigation bugs be found in simulation? An exploratory study," *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2017*, pp. 150–159, 2017.

[21] J. R. B. del Rosario *et al.*, "Modelling and characterization of a maze-solving mobile robot using wall follower algorithm," *Appl. Mech. Mater.*, vol. 446–447, no. November 2013, pp. 1245–1249, 2014.

[22] A. Zarkasi, H. Ubaya, C. Deri Amanda, and R. Firsandaya, "Implementation of ram based neural networks on maze mapping

algorithms for wall follower robot," *J. Phys. Conf. Ser.*, vol. 1196, no. 1, 2019.

[23] I. Elshamarka and A. Bakar Sayuti Saman, "Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm," *Int. J. Comput. Appl.*, vol. 56, no. 5, pp. 8–13, 2012.

[24] S. Tjiharjadi, M. C. Wijaya, and E. Setiawan, "Optimization maze robot using A* and flood fill algorithm," *Int. J. Mech. Eng. Robot. Res.*, vol. 6, no. 5, pp. 366–372, 2017.

[25] R. Balogh and D. Obdržálek, "Using Finite State Machines in Introductory Robotics," *Adv. Intell. Syst. Comput.*, vol. 829, no. April, pp. 85–91, 2019.

[26] R. Hussain, T. Zielinska, and R. Hexel, "Finite state automaton based control system for walking machines," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 3, pp. 1–14, 2019.

[27] Ł. Bienias, K. Szczepański, and P. Duch, "Maze Exploration Algorithm for Small Mobile Platforms," *Image Process. Commun.*, vol. 21, no. 3, pp. 15–26, 2017.

[28] L. K. Li, "Implementation of the Trémaux Maze Solving Algorithm to an Omnidirectional Mobile Robot," in *International Conference on Electronics, Information and Communication*, 2018, pp. 1–3.

[29] Y. F. Hendrawan, "Comparison of Hand Follower and Dead-End Filler Algorithm in Solving Perfect Mazes," *J. Phys. Conf. Ser.*, vol. 1569, p. 022059, 2020.

[30] J. Y. Hii, J. Lee, and Y. Chuah, "Optimization and Simulation of A Navigation Robot in Mazes," *Int. J. Eng. Res. Technol.*, vol. 10, no. 05, pp. 222–227, 2021.