SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I7P118

Volume 11 Issue 7, 208-214, July 2024

Original Article

Optimizing Robot Maze Navigation: A Novel Control Algorithm for Enhancing Energy Efficiency

Thi-Mai-Phuong Dao[1*], Tien-Dung Nguyen[2], Thi-Duyen Bui[2], Quoc-Hoan Tran[2], Van-Tien Nguyen[1], Ngoc-Khoat Nguyen[2]

[1]*Faculty of Electrical Engineering, Hanoi University of Industry, Hanoi, Vietnam.*
[2]*Faculty of Control and Automation, Electric Power University, Hanoi, Vietnam.*

[*]*Corresponding Author* : *daophuong@haui.edu.vn*

Received: 08 May 2024 Revised: 11 June 2024 Accepted: 08 July 2024 Published: 26 July 2024

Abstract *- Robots designed for maze-solving tasks often rely on simple pathfinding strategies, such as consistently following the left or right wall. While effective in basic mazes, these approaches can become inefficient and energy-intensive for more intricate layouts. In the most critical scenario, a robot depleted of energy could become trapped within the maze, rendering its mission unsuccessful. This paper addresses this challenge by introducing a novel algorithm for navigating mazes with the objective of finding the shortest path to the exit. This algorithm goes beyond traditional wall-following methods, aiming to optimize energy consumption and navigation efficiency. Additionally, the paper explores the integration of a PID control algorithm to enhance the robot's movement precision within the maze environment. By maintaining a straight line, when possible, the robot can further minimize unnecessary movements and enhance its overall performance. This work contributes to the field of robotic maze-solving by proposing a more sophisticated pathfinding approach that prioritizes energy conservation and efficient navigation. Furthermore, the implementation of a PID control algorithm allows for improved trajectory control, leading to more precise movements within the maze.*

Keywords *- Maze solving, Robots, Energy efficiency, Optimization, PID.*

1. Introduction

In recent years, various types of maze-solving robots have emerged worldwide. However, these robots typically employ the left-hand or right-hand wall-following method to navigate mazes [1-5]. This approach results in significant energy consumption, particularly in complex mazes, and sometimes robots may run out of power midway through the maze. Consequently, this research focuses on developing a novel robot, termed the Energy-Efficient Maze-Solving Robot, capable of shortening the robot's path during its second traversal of the maze.

Energy-Efficient Maze-Solving Robots (EEMSRs) are a specialized form of intelligent robots designed to autonomously navigate and escape mazes or complex grid systems while minimizing energy consumption. The primary objective of EEMSRs is to enhance maze-solving performance while simultaneously reducing energy expenditure. These robots employ intelligent algorithms to navigate mazes, relying on sensor data and data analysis to make optimal directional decisions. The EEMSRs possess a wide range of practical applications, spanning from industrial settings to scientific research. In industrial environments, they can be utilized for inspection and maintenance of inaccessible sewers, pipes, or ductwork. Within the realm of scientific research, the EEMSRs can facilitate the exploration and investigation of maze structures or other complex environments. The exploration of autonomous maze-solving robots has gained significant traction in recent years. Several research efforts have investigated methodologies for robots to navigate and escape complex maze structures [6-13]. However, a key challenge identified in these studies is the potential for errors during pathfinding due to non-linear robot movements. This is particularly problematic in intricate mazes where precise navigation is crucial.

This research proposes a novel approach to address the limitations of previous maze-solving algorithms. We incorporate the well-established PID (Proportional-Integral-Derivative) control algorithm, a cornerstone of robot control research, into the maze-solving process [14-17]. The PID algorithm offers a robust framework for real-time feedback and error correction, ensuring the robot maintains a more linear and efficient path within the maze.

The current proposed approach integrates the shortest pathfinding method, applicable to any maze configuration, with the PID control system. This combined strategy allows

the robot to not only identify the most efficient route through the maze but also execute the chosen path with greater accuracy and minimal deviations.

Consequently, the proposed algorithm contributes to significant energy optimization for maze-solving robots. By minimizing unnecessary movements and maintaining a more linear path, the robot's overall energy expenditure is reduced, leading to enhanced performance and potentially longer operational times within the maze.

This research builds upon existing knowledge in maze-solving robotics by introducing a novel integration of pathfinding and PID control. The proposed algorithm holds promise for improved efficiency and accuracy in navigating complex mazes, with potential applications in various fields such as search and rescue operations, infrastructure inspection, and autonomous exploration.

2. Algorithm and Method to EEMSRs

2.1. Hardware Configuration of the Robotic Vehicle

One of the most common and applicable types of the EEMSRs is robotic vehicles. The proposed robotic vehicle's structure is illustrated in Figure 1. This vehicle-type robot employs six sensors: Sensor1, Sensor2, Sensor3, Sensor4, Sensor5, and Sensor6.

The robot utilizes DC motors attached to its wheels. An Arduino Mega 2560 serves as the robot's central controller. To navigate around obstacles within the maze and remember the path while stopping at the maze's "exit," the sensors are named and positioned as follows:

- The Infrared sensor Sensor1 is positioned on the left side.
- The Infrared sensor Sensor2 is positioned at the front.
- The Infrared sensor Sensor3 is positioned on the right side.
- The Infrared sensors Sensor4 and Sensor5 are positioned at the rear right side.
- The Infrared sensor Sensor6 is directed downward to detect the black line at the "Exit".

2.2. PID - Based Straight-Line Control Method

The proposed wall-following algorithm utilizes two sensors, Sensor4 and Sensor5, mounted on the robot's right side. Sensor5 is set to be more sensitive than Sensor4. Assuming Sensorx = "0" indicates wall detection and Sensorx = "1" indicates no wall (where *x* ranges from 1 to 6), three scenarios can occur:

1. If Sensor5 = 0 and Sensor4 = 0: Instruct the robot to turn left (error = -2).
2. If Sensor5 = 0 and Sensor4 = 1: Instruct the robot to move forward (error = 0).
3. If Sensor5 = 1 and Sensor4 = 1: Instruct the robot to turn right (error = 2).

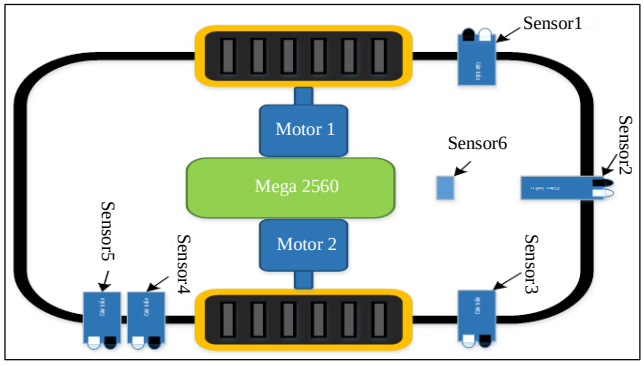

Fig. 1 Configuration of the robotic vehicle

Following the above algorithm, the PID (Proportional-Integral-Derivative) control algorithm is employed to calculate and adjust the robot's movement for straight-line navigation.

2.3. Right - Wall Following Algorithm to Solve the Maze

This study proposes a right-wall following algorithm for maze-solving by a robotic vehicle. The specific algorithm is as follows:

```
{
P = error;
if(leftMotorSpeed!=maxspeed||rightMotorSpeed!=maxspeed)
{I = I + previousI;previousI=I;}
D = error-previousError;
PIDvalue = (Kp*P) + (Ki*I) + (Kd*D);
previousError = error;
}
void PID_write_speed() //Control of the robot vehicle
{
leftMotorSpeed = min_speed + PIDvalue;
rightMotorSpeed = min_speed - PIDvalue;
leftMotorSpeed=constrain(round(leftMotorSpeed), 0, maxspeed);
rightMotorSpeed=constrain(round(rightMotorSpeed), 0, maxspeed);
analogWrite(ENA,leftMotorSpeed);analogWrite(ENB,rightMotorSpeed);
}
```

Only one initial exploration is required to solve the maze using the right-wall following approach. In subsequent moves, the robot will proceed directly to the goal.

During the maze-solving process, the robot may encounter the following scenarios (see Figure 2).

- Case 1. $Sensor_1$==0 && $Sensor_2$==0 && $Sensor_3$==0 => Dead-end street (NO)
- Case 2. $Sensor_1$==0 && $Sensor_2$==1 && $Sensor_3$==0 => Only forward path (U)

- Case 3. $Sensor_1$==1 && $Sensor_2$==1 && $Sensor_3$==0 => Left and Forward Paths (LU)
- Case 4. $Sensor_1$==0 && $Sensor_2$==1 && $Sensor_3$==1 => Right and Forward Paths (ru)
- Case 5. $Sensor_1$==0 && $Sensor_2$==0 && $Sensor_3$==1 => only right Path (r)
- Case 6. $Sensor_1$==1 && $Sensor_2$==0 && $Sensor_3$==0 => only Left Path (l)
- Case 7. $Sensor_1$==1 && $Sensor_2$==1 && $Sensor_3$==1 => Three Paths (lur)
- Case 8. $Sensor_1$==1 && $Sensor_2$==0 && $Sensor_3$==1 => Left and Right Paths (lr)
- Case 9. $Sensor_6$==0 (Detecting Door Exit) (Ignore Sensors Sensor1 to Sensor5) => Prioritize Stopping the Vehicle

Applying the above algorithm, the procedure to solve the maze is proposed in this work. In theory, an array-based data structure is employed to store the scenarios encountered by the robotic vehicle sequentially. This information is subsequently utilized to optimize the path for the next traversal. The specific scenarios and their corresponding actions are as follows:

Case 1: Turn Back ("Turn around"): In this scenario, the robotic vehicle executes a 180-degree turn, effectively reversing its direction.

Case 2: Move Straight: The robotic vehicle maintains its current heading and continues moving forward in a straight line.

Case 3: Move Straight: Similar to Case 2, the robotic vehicle maintains its current heading and continues moving forward in a straight line.

Case 4: Turn Right: The robotic vehicle executes a 90-degree turn to the right, altering its direction accordingly.

Case 5: Turn Right: Like Case 4, the robotic vehicle executes a 90-degree turn to the right, altering its direction accordingly.

Case 6: Turn Left: The robotic vehicle executes a 90-degree turn to the left, altering its direction accordingly.

Case 7: Turn Right: Similar to Case 4, the robotic vehicle executes a 90-degree turn to the right, altering its direction accordingly.

Case 8: Turn Right: Like Case 4, the robotic vehicle executes a 90-degree turn to the right, altering its direction accordingly.

Case 9: Stop the Robotic Vehicle: The robotic vehicle ceases all movements and comes to a complete halt.

While it may seem that multiple cases involve turning right (Cases 4, 5, 7, and 8), the differentiation between these cases is crucial for accurate maze navigation.

Each case represents a distinct scenario that requires a specific response from the robotic vehicle.

Case 4: This case specifically addresses situations where the vehicle encounters a right turn while maintaining a clear path ahead.

Case 5: This case handles scenarios where the vehicle encounters a right turn and an obstacle on the left side.

Case 6: This case addresses situations where the vehicle encounters a left turn and an obstacle on the right side.

Case 7: This case handles scenarios where the vehicle encounters a right turn and an obstacle directly ahead.

Case 8**:** This case addresses situations where the vehicle encounters a right turn and an obstacle on both the left and ahead sides.

By differentiating between these cases, the algorithm ensures that the robotic vehicle makes informed decisions and navigates the maze efficiently while avoiding collisions and unintended actions.

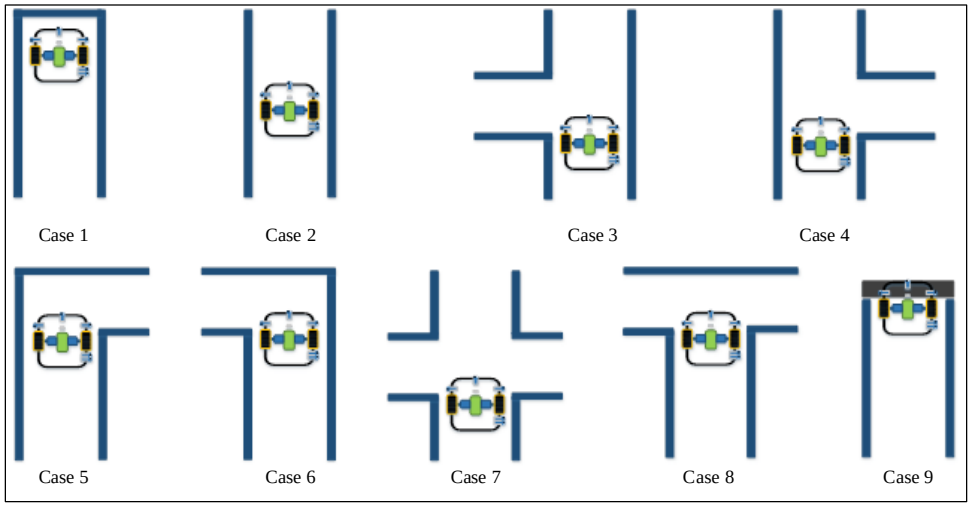

Fig. 2 Potential cases encountered in maze navigation

2.4. Recording Possible Cases

To optimize the maze-solving process and minimize unnecessary movements, a case recording mechanism is implemented. This mechanism involves storing the sequence of scenarios encountered by the robotic vehicle as it traverses the maze. The recorded cases serve as a reference for subsequent traversals, allowing the vehicle to make informed decisions and avoid revisiting previously explored paths.

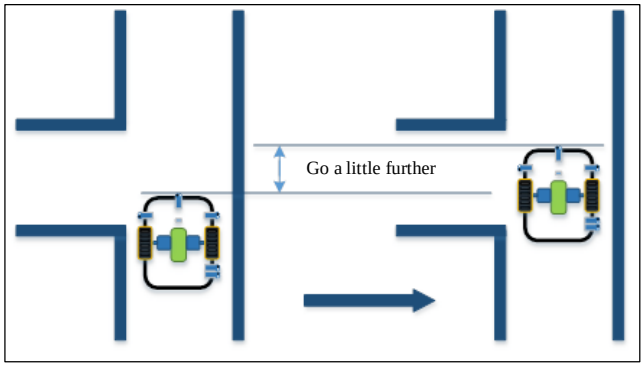

Fig. 3 Illustrating robot's straight-line movement

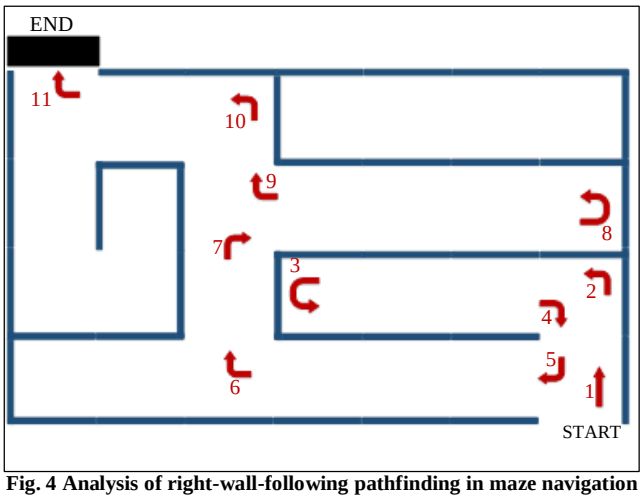

Fig. 4 Analysis of right-wall-following pathfinding in maze navigation

Consider the simplified maze depicted in Figure 4. As the robotic vehicle navigates through this maze, it encounters various scenarios, such as dead ends, intersections, and open paths. A specific case represents each of these scenarios. These cases are crucial for optimizing the maze-solving process and minimizing unnecessary movements.

Starting from the initial position, the robotic vehicle traverses the maze and encounters the following cases:

- Case 1: Move Straight (S): The vehicle moves forward along the initial path.
- Case 2: Turn Left (L): The vehicle encounters an intersection and turns left.
- Case 3: Turn Around (B): The vehicle reaches a dead end and turns around.
- Case 4: Turn Right (R): The vehicle encounters an intersection and turns right.
- Case 5: Turn Right (R): The vehicle continues along the right path.
- Case 6: Turn Right (R): The vehicle encounters another right turn.
- Case 7: Turn Right (R): The vehicle continues along the right path.
- Case 8: Turn Around (B): The vehicle reaches a dead end and turns around.
- Case 9: Turn Right (R): The vehicle encounters an intersection and turns right.
- Case 10: Turn Left (L): The vehicle encounters another intersection and turns left.
- Case 11: Turn Right (R): The vehicle reaches its destination.

To effectively store and utilize the recorded cases, an array data structure is employed. The array is initialized as an empty array:

mode[]={};

As the robotic vehicle progresses through the maze, each encountered case is appended to the array. For instance, after completing the traversal of the maze in Figure 4, the array would contain the following sequence of cases:

mode[]={SLBRRRRBRLR};

This recorded case sequence serves as a valuable reference for subsequent traversals, allowing the vehicle to make informed decisions and avoid revisiting previously explored paths.

2.5. Path Optimization Algorithm

To further enhance the maze-solving efficiency, a path optimization algorithm is employed. This algorithm utilizes a set of rules to reduce the number of unnecessary turns and movements, thereby optimizing the overall path traversal.

2.5.1. Optimization Rules

The path optimization algorithm is based on the following rules:

RBL = B: If the sequence of cases is “Right Turn - Back Turn - Left Turn,” then the sequence can be simplified to “Back Turn.”

RBS = L: If the sequence of cases is “Right Turn - Back Turn - Straight,” then the sequence can be simplified to “Left Turn.”

LBR = B: If the sequence of cases is “Left Turn - Back Turn - Right Turn,” then the sequence can be simplified to “Back Turn.”

SBR = L: If the sequence of cases is “Straight - Back Turn - Right Turn,” then the sequence can be simplified to “Left Turn.”

SBS = B: If the sequence of cases is “Straight - Back Turn - Straight,” then the sequence can be simplified to “Back Turn.”

RBR = S: If the sequence of cases is “Right Turn - Back Turn - Right Turn,” then the sequence can be simplified to “Straight.”

2.5.2. Path Optimization Application

To illustrate the application of the path optimization algorithm, consider the recorded case sequence from Figure 4: mode[]={SLBRRRRBRLR};

Applying the optimization rules, we can simplify the sequence as follows:

SLBRRRRBRLR => SBRRRRBRLR (RBL = B)

SBRRRRBRLR => LRRRBRLR (SBR = L)

LRRBRLR => LRSLR (RBR = S)

The resulting optimized case sequence is:

mode[]={LRSLR};

This optimized sequence represents the most efficient path through the maze, minimizing unnecessary turns and movements.

2.5.3. Visualization of Optimized Path

Figure 5 depicts the robotic vehicle traversing the maze using the optimized path. As evident from this figure, the vehicle follows a more direct and efficient route compared to the original path.

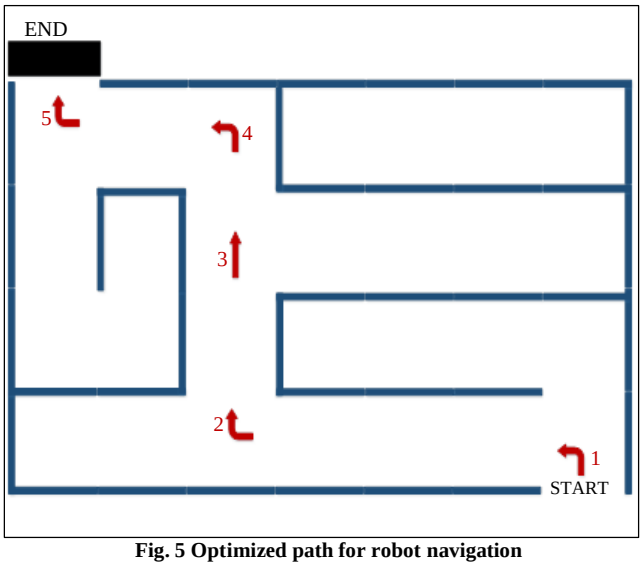

Fig. 5 Optimized path for robot navigation

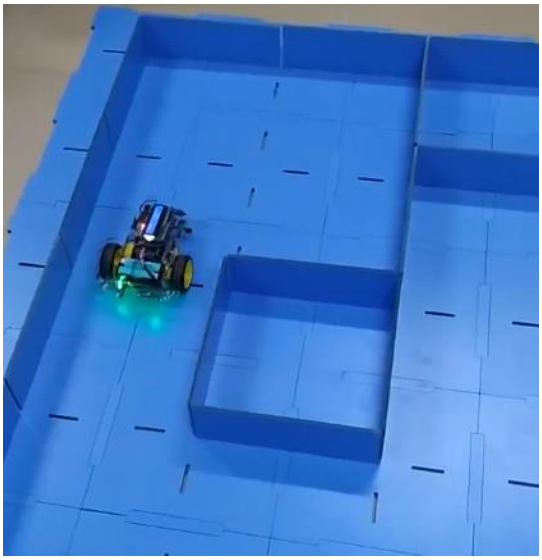

Fig. 6 An illustration of robot testing with the maze

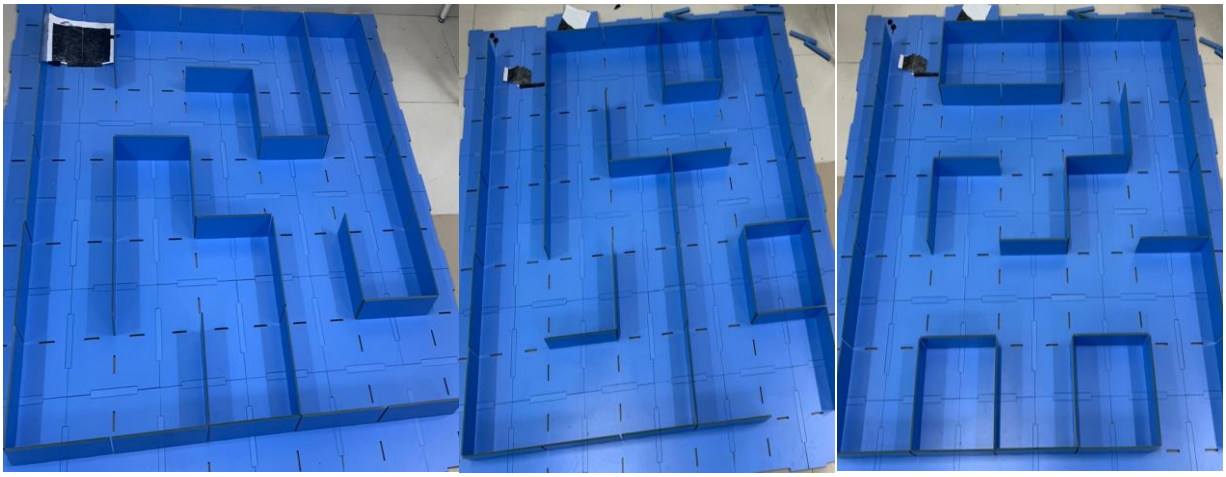

Fig. 7 Three maze scenarios were solved in this study

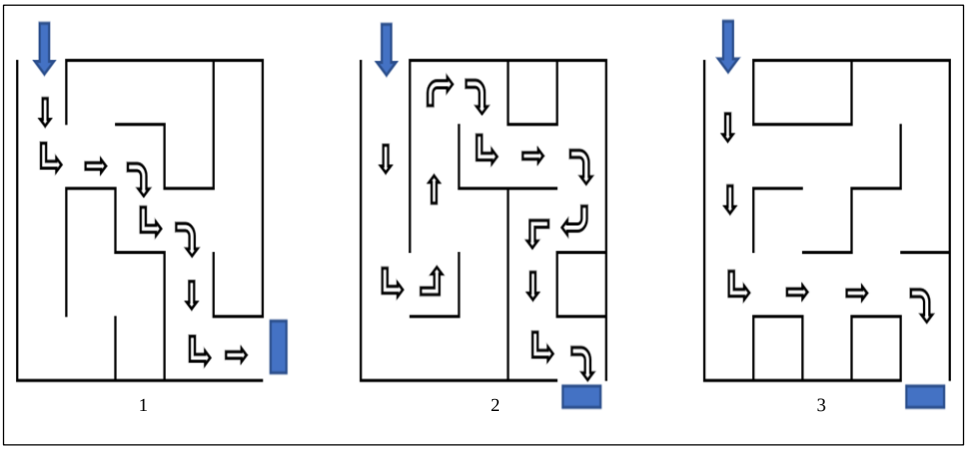

Fig. 8 Replicating optimized robot paths for three mazes from Figure 7

3. Experiments and Discussions

3.1. Maze Testing Illustration

Figure 6 captures an image of the robotic vehicle undergoing maze testing. The vehicle is visible, navigating through a maze structure, demonstrating its ability to handle various path configurations and obstacles. To comprehensively evaluate the proposed maze-solving algorithm, a diverse set of mazes is carefully selected for testing purposes. These mazes are designed to present a range of challenges and complexities, allowing for a thorough assessment of the algorithm's performance in various scenarios. Figure 7 showcases three representative mazes selected for the experimental evaluation. These mazes (from left to right sides) exhibit varying levels of complexity, with the first maze being relatively simple, the second maze introducing more intricate path configurations, and the third maze featuring multiple dead ends and obstacles.

3.2. Performance Evaluation: Maze Solving Times

Table 1 presents the quantitative results for the time taken to solve the three selected mazes. The table compares the performance of the maze-solving algorithm with and without path optimization. As evident from the table, the time taken to solve each maze using the optimized algorithm is significantly lower compared to the time taken when only right turns are employed. This substantial reduction in solving time highlights the effectiveness of the proposed path optimization technique. The optimized algorithm achieves an average time reduction of 36.86% across the three mazes. This implies that the robotic vehicle utilizing the optimized algorithm can traverse the mazes approximately 61.36% faster compared to the traditional right-turn-only approach. This significant improvement in performance demonstrates the superiority of the proposed maze-solving algorithm.

By minimizing unnecessary turns and movements, the algorithm enables the robotic vehicle to navigate the mazes more efficiently, saving energy and time. The experimental results provide compelling evidence that the proposed path optimization algorithm offers a substantial advantage in terms of maze-solving efficiency. This finding contributes to the advancement of robotic maze-solving techniques and paves the way for further research and development in this area.

Table 1. Quantitative experiment results

Maze	Time to Solve (Only Right Turns)	Time to Solve (Optimized Algorithm)	Time Reduction
Maze 1	12.05 seconds	5.22 seconds	57.35%
Maze 2	13.23 seconds	11.25 seconds	15.22%
Maze 3	9.35 seconds	4.98 seconds	46.75%
Total Time	34.63 seconds	21.25 seconds	38.94%

4. Conclusions and Future Work

4.1. Summary of Findings

The experimental evaluation of the proposed maze-solving algorithm with path optimization demonstrates its effectiveness in accurately navigating and solving diverse mazes. The algorithm consistently outperforms the traditional right-turn-only approach, achieving significant time reductions and minimizing unnecessary movements.

4.2. Contributions to the Field

The research presented in this paper contributes to the advancement of robotic maze-solving techniques by introducing an efficient path optimization algorithm that enhances the robot's performance in terms of accuracy, efficiency, and energy conservation. This work highlights the potential of such algorithms in enabling robots to operate more effectively and autonomously in complex environments.

4.3. Future Directions

While the proposed algorithm demonstrates promising results, further research and development are warranted to expand its capabilities and address potential limitations. One

area of focus could be the development of strategies for the robot to explore the entire maze and identify the shortest overall path to the exit. This would require incorporating techniques for obstacle detection and path planning that consider the entire maze structure. Additionally, exploring alternative optimization algorithms and control strategies could lead to further improvements in the robot's performance. Investigating the application of these techniques to different types of mazes and environments would also be valuable for enhancing the generalizability of the approach.

4.4. Overall Significance

The proposed maze-solving algorithm with path optimization represents a significant step forward in the development of autonomous robots capable of navigating complex environments.

Its ability to improve efficiency, reduce energy consumption, and enhance accuracy holds promise for a wide range of applications, including search and rescue operations, exploration missions, and industrial automation.

References

[1] Shatha Alamri et al., "Autonomous Maze Solving Robotics: Algorithms and Systems," *International Journal of Mechanical Engineering and Robotics Research*, vol. 10, no. 12, pp. 668-675, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Kuo-Chi Chang et al., "Shortest Distance Maze Solving Robot," *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*, Dalian, China, pp. 283-286, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Rares Covaci, Gabriel Harja, and Ioan Nascu, "Autonomous Maze Solving Robot," *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, Cluj-Napoca, Romania, pp. 1-4, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Rahul Kumar et al., "Maze Solving Robot with Automated Obstacle Avoidance," *Procedia Computer Science*, vol. 105, pp. 57-61, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[5] Shatha Alamri et al., "An Autonomous Maze-Solving Robotic System Based on an Enhanced Wall-Follower Approach," *Machines*, vol. 11, no. 2, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] A. Bemporad, M. Di Marco, and A. Tesi, "Wall-Following Controllers for Sonar-Based Mobile Robots," *Proceedings of the 36th IEEE Conference on Decision and Control*, San Diego, USA, vol. 3, pp. 3063-3068, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[7] Irina Cojuhari, "Algorithm for Self-Tuning the PID Controller," *Journal of Engineering Sciences*, vol. 28, no. 4, pp. 63-73, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[8] Swati Mishra, and Pankaj Bande, "Maze Solving Algorithms for Micro Mouse," *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, Bali, Indonesia, pp. 86-93, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[9] Min Raj Nepali et al., "A Novel Wall Following Algorithm for Mobile Robots," *International Journal of Robotics and Automation*, vol. 5, no. 2, pp. 15-23, 2014. [Google Scholar] [Publisher Link]

[10] Xin Wei et al., "A Wall-Following Algorithm Based on Dynamic Virtual Walls for Mobile Robots Navigation," *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Okinawa, Japan, pp. 46-51, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[11] Mohammad O.A. Aqel et al., "Intelligent Maze Solving Robot Based on Image Processing and Graph Theory Algorithms," *2017 International Conference on Promising Electronic Technologies (ICPET)*, Deir El-Balah, Palestine, pp. 48-53, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[12] Hind Zuhair Khaleel, and Bashra Kadhim Oleiwi, "Ultrasonic Sensor Decision-Making Algorithm for Mobile Robot Motion in Maze Environment," *Bulletin of Electrical Engineering and Informatics*, vol. 13, no. 1, pp. 109-116. 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Indar Sugiarto, Lauw Lim Un Tung, and Mohammad Ismail Rahman, "Implementation of Fuzzy Logic in FPGA for Maze Tracking of a Mobile Robot Based on Ultrasonic Distance Measurement," *Jurnal Teknik Elektro*, vol. 9, no. 1, pp. 96-102, 2010. [Google Scholar] [Publisher Link]

[14] Bot Reboot, Line Follower Robot with PID controller, hackster.io, 2020. [Online]. Available: https://www.hackster.io/anova9347/line-follower-robot-with-pid-controller-cdedbd

[15] Shijie Guo, "PID Control Based on Machine Learning Algorithm in Robot Path Optimization," *Proceedings of the 2023 International Conference on Mechatronics and Smart Systems*, vol. 12, pp. 284-291, 2023. [CrossRef] [Publisher Link]

[16] Rongchang Liu, "Research on Control and Optimization of Robot's Bottom Action Based on PID Algorithm," *2023 International Conference on Power, Electrical Engineering, Electronics and Control (PEEEC)*, Athens, Greece, pp. 699-703, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Abdullah Al Mamun Khan et al., "Design and Implementation of a Robot for Maze-Solving with Turning Indicators Using PID Controller," *2013 International Conference on Informatics, Electronics & Vision (ICIEV)*, Dhaka, Bangladesh, pp. 1-6, 2013. [CrossRef] [Google Scholar] [Publisher Link]