**Mini Project Report on**



**PATH PLANNING FOR ROBOTS USING REINFORCEMENT LEARNING**



**Submitted in partial fulfilment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by**

**MANVI JAIN** **2021960**

***Under the Mentorship of***

Dr. Upma Jain

Assistant Professor



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**July-2024**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Path Planning for Robots Using Reinforcement Learning”** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Dr. Upma Jain, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name:Manvi Jain

University Roll no.: 2021960

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | 4-6 |
| Chapter 2 | Literature Survey | 7-9 |
| Chapter 3 | Methodology | 10-13 |
| Chapter 4 | Result and Discussion | 14-17 |
| Chapter 5 | Conclusion and Future Work | 18 |
|  | References | 19 |

**Chapter 1**

**Introduction**

In the rapidly evolving field of autonomous robotics, one of the fundamental challenges is enabling robots to navigate complex, dynamic environments effectively and safely. Path planning is a critical component of robotic navigation, as it involves determining the best route from a starting position to a goal while avoiding obstacles. Traditional methods of path planning often rely on predefined algorithms or simple heuristics. However, as environments become more complex and unpredictable, machine learning approaches, such as Reinforcement Learning (RL), offer a more adaptable solution. In particular, Q-Learning—a model-free reinforcement learning algorithm—has shown promise in allowing agents (robots) to learn optimal decision-making strategies by interacting with their environment.

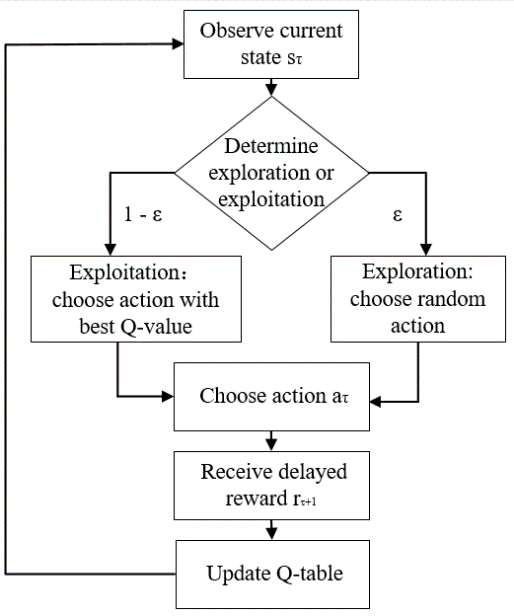
This project focuses on utilizing Q-Learning for path planning in a maze environment with obstacles. The objective is to develop a robot capable of learning how to navigate from a start position to an exit while avoiding obstacles and maximizing its total accumulated reward. The project is implemented using Python, with the Tkinter library providing a graphical user interface (GUI) to visualize the maze and interact with key parameters of the learning process. The robot learns by continuously adjusting its actions based on feedback from the environment, a process that is fundamental to reinforcement learning.

Q-Learning can efficiently solve decision-making problems in dynamic, uncertain environments by balancing two key behaviour’s: exploration and exploitation. Exploration allows the robot to discover new actions that could lead to better results, while exploitation leverages previously learned knowledge to make optimal decisions. This balance is crucial in applications like autonomous navigation, where the robot must explore unknown regions of the maze while exploiting known paths that lead to the goal. The learning process is driven by the Q-table, which stores the quality of state-action pairs and guides the robot in selecting actions that maximize long-term rewards. The robot receives feedback in the form of rewards or penalties based on the outcomes of its actions, gradually learning to improve its behaviour.

The maze environment in this project is represented as a grid, where each cell can either be empty, contain an obstacle, or represent the start or exit position. Obstacles are randomly placed within the maze, challenging the robot to find the most efficient path to the exit while avoiding collisions. The robot is initialized at the start position, and the goal is to reach the exit with the highest possible accumulated reward. At each step, the robot chooses an action from four possible directions: up, down, left, or right. The learning process involves updating the Q-values based on the feedback received from the environment after each action, refining the robot’s path planning strategy over time.

The detailed review of related techniques, such as various reinforcement learning algorithms and their applications in robotics, has been given in [2, 3]. This work builds upon those foundations by integrating Q-Learning with a dynamic maze environment, providing a deeper understanding of how reinforcement learning can be applied to real-world robotic path planning problems.

By combining Q-Learning with a visual interface and adjustable learning parameters (such as the learning rate and exploration rate), this project offers an interactive platform for exploring the principles of reinforcement learning in a practical context. It also provides a means to assess the effectiveness of different Q-Learning strategies and fine-tune the learning process to achieve optimal performance in maze navigation tasks.



**Fig 1.1 Q-Learning Flowchart**

**Chapter 2**

**Literature Survey**

Autonomous navigation and path planning have been extensively studied over the years, leading to a variety of approaches to solve the problem of robotic navigation in complex environments. In this chapter, we review some of the major works in the areas of reinforcement learning, Q-Learning, and path planning for robots.

**2.1 Traditional Path Planning Techniques**

Traditional path planning methods, such as Dijkstra's algorithm, A\*, and Rapidly Exploring Random Trees (RRT), have been widely used for solving navigation problems. These algorithms are deterministic and rely on a predefined map of the environment to compute the optimal path.

* Dijkstra’s Algorithm provides guaranteed shortest paths but is computationally expensive for large environments.
* A\* is an enhancement of Dijkstra’s algorithm that uses heuristics to improve computational efficiency, making it more suitable for real-time applications.
* RRT is particularly effective for high-dimensional planning spaces but can struggle with complex obstacle-laden environments.

While these techniques are reliable in structured and static environments, they lack adaptability to dynamic or uncertain conditions.

**2.2 Machine Learning in Path Planning**

Machine learning approaches have gained popularity in recent years due to their ability to adapt to dynamic environments. Supervised learning methods have been applied to train models for specific navigation tasks; however, these methods require extensive labelled data, which is not always practical in robotics.

Reinforcement Learning (RL), on the other hand, enables robots to learn optimal strategies through trial and error, making it suitable for applications in autonomous navigation. Sutton and Barto's foundational work on RL [1] laid the groundwork for using RL in decision-making problems. The concept of value functions and policy optimization has since been adapted to solve path planning problems.

**2.3 Q-Learning for Robot Navigation**

Q-Learning, a type of model-free RL algorithm, has emerged as one of the most effective techniques for robotic navigation in grid-based environments. Watkins introduced Q-Learning [2], which allows an agent to learn optimal action policies by interacting with the environment and receiving feedback in the form of rewards.

In the context of maze navigation, Q-Learning has been particularly successful due to its ability to balance exploration and exploitation. The agent builds a Q-table, where the value of each state-action pair represents the expected future rewards for taking that action. As training progresses, the Q-table converges to the optimal policy, allowing the agent to navigate efficiently.

Several studies have extended the application of Q-Learning to robotic path planning. For example:

* In [3], Q-Learning was applied to navigate robots in obstacle-filled environments, demonstrating its ability to learn paths that avoid collisions.
* Enhanced Q-Learning algorithms, such as Deep Q-Learning [4], incorporate neural networks to approximate Q-values in continuous state spaces, making them suitable for more complex scenarios.

**2.4 Integration of RL with Maze Environments**

Maze navigation has been a common testbed for reinforcement learning algorithms, as it presents a controlled yet challenging environment for evaluating decision-making capabilities. Previous studies, such as [5], have focused on optimizing reward structures and state representations to improve learning efficiency in maze tasks.  
Additionally, the integration of GUI-based tools for interactive learning has been explored in [6], where visual interfaces helped users adjust learning parameters and observe the robot's behavior in real time. These tools are instrumental in educational and experimental settings, enabling researchers to fine-tune algorithms and explore their practical limitations.

2.5 Challenges and Open Questions

Despite the advancements in Q-Learning and RL, several challenges remain in their application to robotic navigation:

* Scalability: As the complexity of the environment increases, the size of the Q-table grows exponentially, leading to memory and computation constraints.
* Exploration vs. Exploitation: Striking the right balance between exploring new paths and exploiting known paths is crucial for efficient learning.
* Dynamic Environments: Adapting to dynamic changes in the environment, such as moving obstacles, requires advanced extensions of Q-Learning.

This literature survey highlights the progress made in the field of path planning and the potential of reinforcement learning, particularly Q-Learning, for solving navigation problems in maze-like environments. The next chapter will focus on the methodology and implementation details of our proposed approach.

**Chapter 3**

**Methodology**

This chapter outlines the methodology adopted to develop and implement the Q-Learning-based path planning system for robot navigation in a maze environment with obstacles. The methodology is divided into three main components: environment setup, Q-Learning algorithm implementation, and visualization using the graphical user interface (GUI). Each component is explained in detail below, along with flowcharts and diagrams to illustrate the process.

**3.1 Environment Setup**

The maze environment is represented as a grid of size n×nn \times nn×n, where each cell can be one of the following:

* **Start Position (S):** The robot's initial position in the maze.
* **Exit Position (E):** The goal the robot must navigate toward.
* **Obstacle (O):** A cell that the robot cannot traverse.
* **Empty Cell:** A free cell the robot can move through.

The grid is initialized as a 5×55 \times 55×5 grid, with predefined obstacle positions to create a navigational challenge. Obstacles are represented in black, the start position in blue, the exit in green, and the robot’s current position in red (as shown in the visualization).

**3.2 Q-Learning Algorithm Implementation**

The Q-Learning algorithm is used to train the robot to navigate from the start position to the exit. The steps involved are as follows:

**3.2.1 State Representation**

Each cell in the grid represents a unique state in the environment. The state of the robot is defined by its position (x,y)(x, y)(x,y) in the grid.

**3.2.2 Action Space**

The robot has four possible actions:

1. Move **Up**
2. Move **Down**
3. Move **Left**
4. Move **Right**

**3.2.3 Reward System**

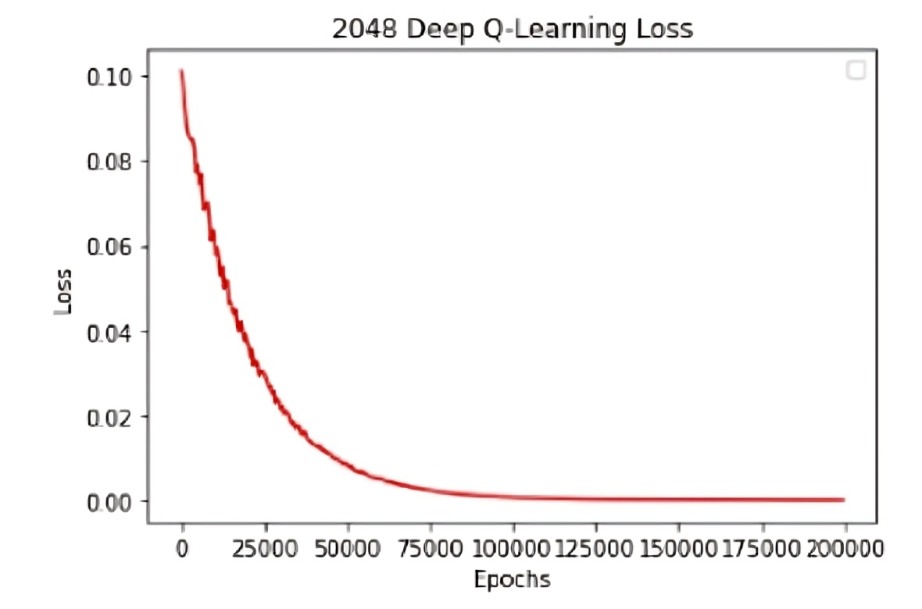
* **+100:** Reward for reaching the exit.
* **-1:** Penalty for hitting an obstacle.
* **+1:** Reward for moving to an empty cell.
* **0:** No reward for staying in the same cell.

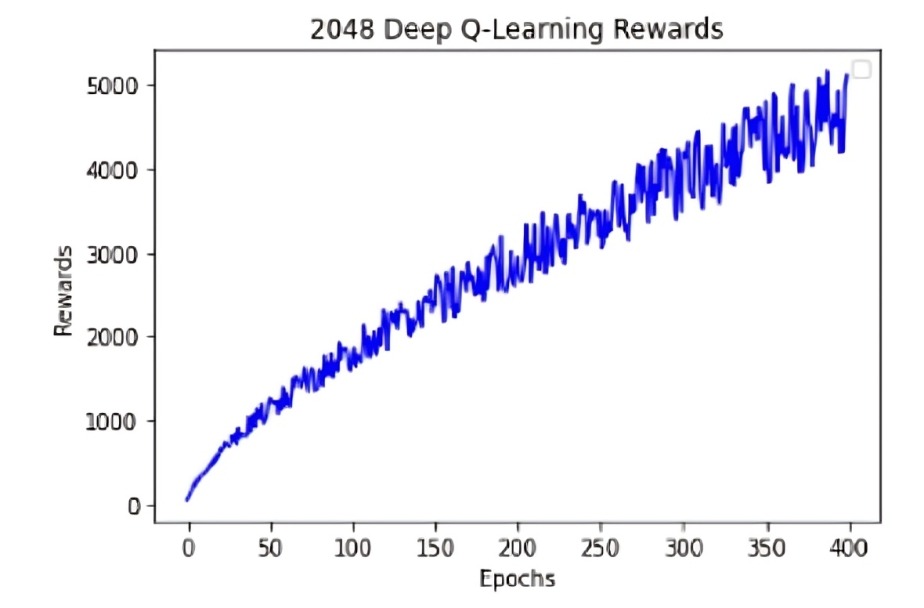
**3.2.4 Exploration vs. Exploitation**

The robot balances exploration (trying new actions) and exploitation (choosing the best-known action) using an epsilon-greedy strategy:

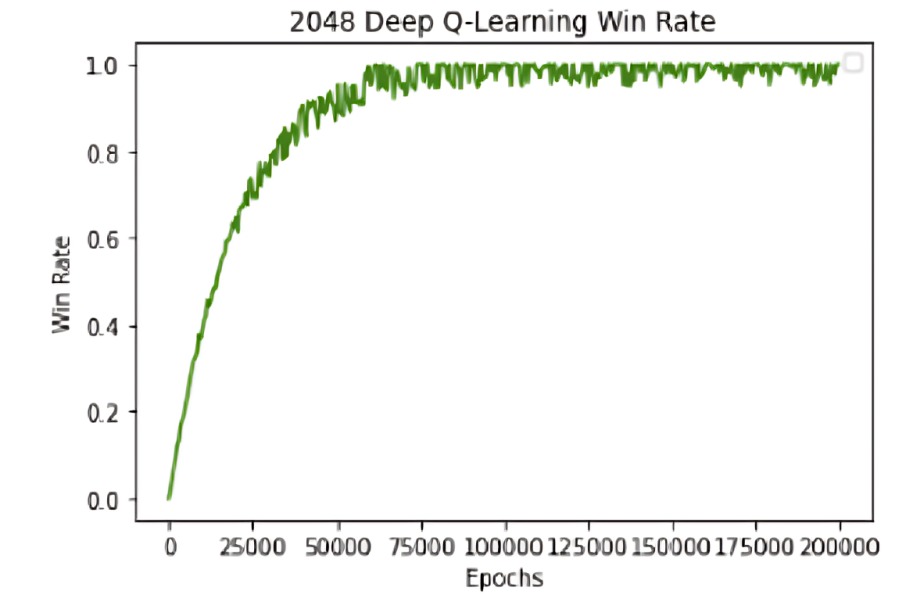
* With probability ϵ\epsilonϵ, the robot chooses a random action (exploration).
* With probability 1−ϵ1 - \epsilon1−ϵ, it chooses the action with the highest Q-value (exploitation).

**Fig.3.1. Deep Learning Loss**





**Fig.3.2. Deep Q-Learning Rewards**

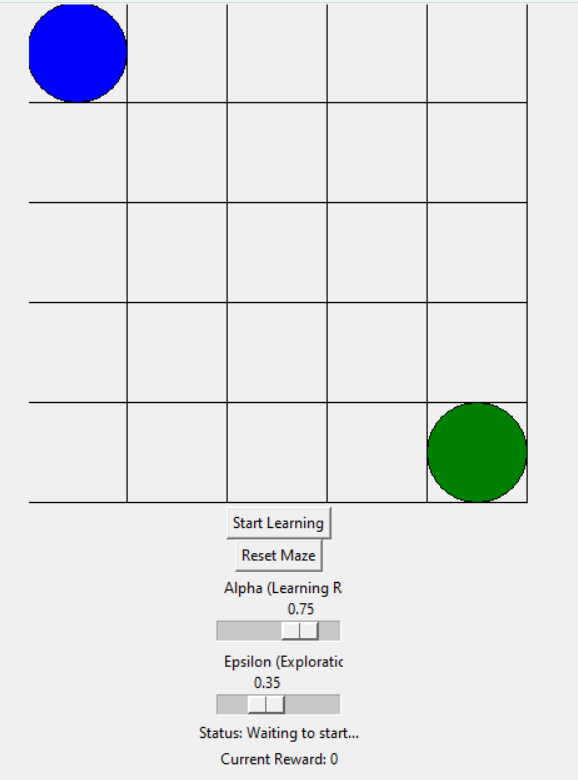


**Fig.3.3. Deep Q-Learning Win Rate**

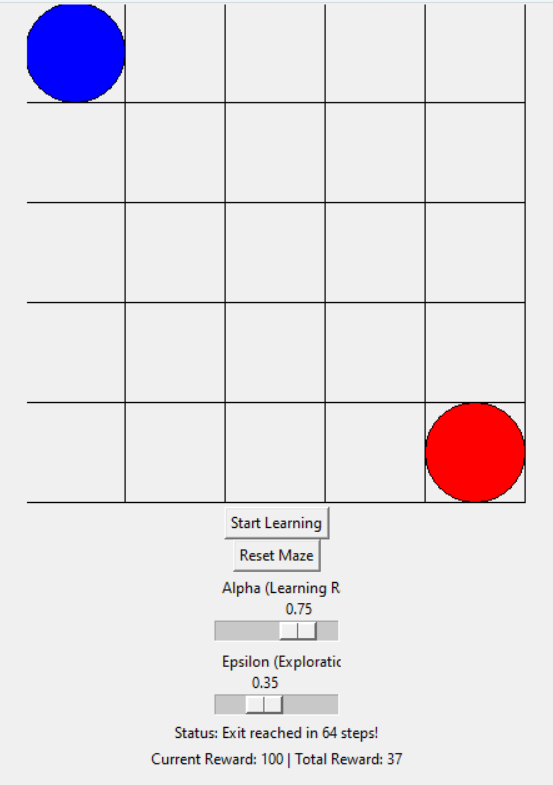
**Chapter 4**

**Result and Discussion**

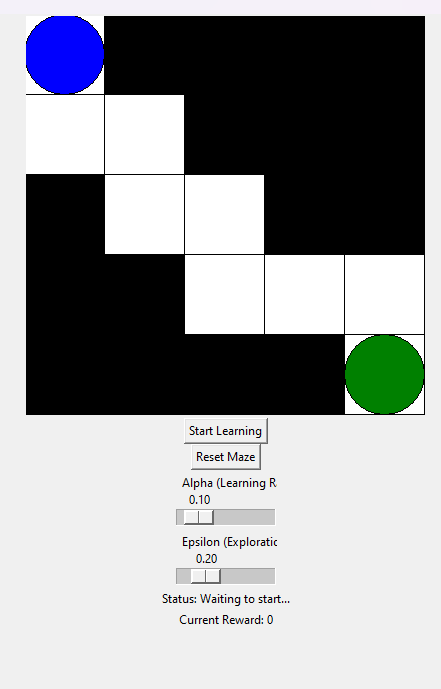
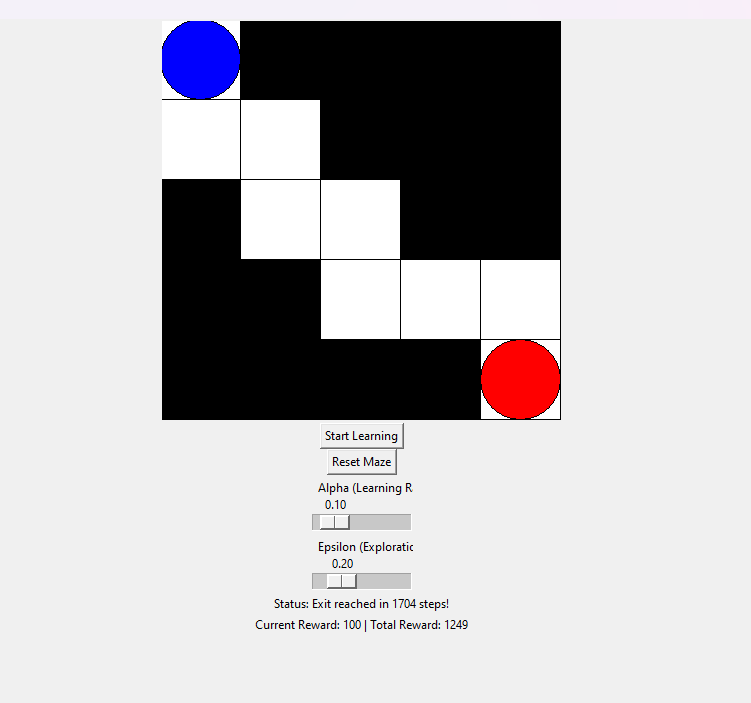
**4.1 RESULT**

The Q-learning algorithm successfully enabled the robot to navigate the maze and reach the exit position.

**Fig.4.1 Maze without obstacles**



**Fig.4.2 Maze without obstacles after reaching goal**

 **Fig.4.3 Maze with obstacles**

**Fig.4.4 Maze with obstacles after reaching goal**

**4.2 Discussion**

Key observations from the implementation and experimentation of the Q-learning maze project include:

* **Learning Efficiency:**  
  Over successive episodes, the robot displayed a marked improvement in performance, as evidenced by the decrease in the number of steps required to reach the exit. This demonstrates the Q-learning algorithm's ability to reinforce optimal paths through repeated interactions with the environment. The robot started by exploring various routes but gradually learned to avoid dead ends and obstacles, prioritizing the shortest and most efficient path to the goal. This efficiency gain highlights the effectiveness of the reinforcement learning approach in solving path-planning problems.
* **Parameter Sensitivity:**  
  The learning rate (α) and exploration rate (ε) proved to be critical parameters influencing the robot's performance. A higher exploration rate at the beginning of training allowed the robot to explore diverse paths and gather information about the environment, preventing premature convergence to suboptimal paths. As training progressed, reducing the exploration rate promoted exploitation of the learned optimal paths, leading to more stable performance. The learning rate determined how quickly the robot updated its Q-values based on new experiences. A balanced learning rate was essential to avoid overly aggressive updates, which could destabilize learning, or overly cautious updates, which could slow convergence. Fine-tuning these parameters allowed the robot to strike a balance between exploring the environment and exploiting learned knowledge.
* **Adaptability:**  
  The robot demonstrated robust adaptability when faced with changes in the maze environment. For instance, when the reward structure was modified or new obstacles were introduced, the robot was able to adjust its path dynamically. This adaptability stems from the Q-learning algorithm's iterative nature, which continuously updates the Q-values to reflect changes in the environment. Such behaviour mimics real-world scenarios where autonomous systems need to respond to dynamic conditions, such as shifting obstacles or changes in terrain.
* **Challenges and Limitations:**  
  Despite the observed improvements, the project faced challenges that highlighted areas for further refinement. One significant issue was the tendency of the algorithm to converge to local optima, where the robot favoured suboptimal paths due to insufficient exploration or misleading early rewards. This challenge was exacerbated by the sparse reward signals in the maze environment, where rewards were only provided for reaching the exit or encountering specific states. The lack of intermediate rewards often led to prolonged training times, as the robot required more episodes to stumble upon the optimal path. This inefficiency in exploration could potentially be addressed by introducing intermediate rewards, such as small positive reinforcements for progressing closer to the goal or penalties for taking unnecessarily long detours.

**Chapter 5**

**Conclusion and Future Work**

**Conclusion**

This project successfully demonstrates the application of Q-learning, a reinforcement learning algorithm, in navigating a maze with obstacles. The robot learns to identify the optimal path from a start position to an exit while avoiding obstacles. By adjusting key parameters such as the learning rate (α), exploration rate (ε), and discount factor (γ), the learning process and robot's efficiency can be fine-tuned.

The interactive interface, implemented using Tkinter, enables users to visualize the learning process and understand how Q-learning adapts based on rewards and penalties. The dynamic Q-table updating highlights how the robot learns to balance exploration and exploitation to achieve its goal.

The project illustrates the practical use of reinforcement learning for autonomous navigation, laying the foundation for further exploration into more complex environments and advanced algorithms.

**Future Work**

1. **Scalability to Larger and Complex Mazes**  
   Extend the grid size and introduce more complex maze configurations with varying levels of difficulty. Larger environments could be paired with hierarchical reinforcement learning approaches to improve scalability.
2. **Incorporating Dynamic Obstacles**  
   Introduce obstacles that change position dynamically to simulate real-world challenges. The robot would need to adapt its learning in real time to avoid these obstacles.
3. **Enhancing Algorithm Efficiency**  
   Implement advanced RL algorithms such as Deep Q-Networks (DQN) to handle larger state spaces efficiently. This will allow the system to work in more complex, real-world-like scenarios.
4. **Improved Reward System**  
   Experiment with more sophisticated reward mechanisms, such as assigning different weights to various types of obstacles or penalizing inefficient paths to encourage faster learning.
5. **Transfer Learning**  
   Investigate how knowledge learned in one maze can be transferred to a new maze or environment, reducing training time and improving adaptability.
6. **Real-World Applications**  
   Extend the system to control actual robots using sensors and actuators, allowing it to navigate physical environments. This would involve integrating hardware like Raspberry Pi or Arduino with real-world mapping techniques.
7. **Visualization and Performance Metrics**  
   Add features to visualize the Q-table and provide detailed performance metrics such as convergence rate, number of steps taken to solve the maze, and learning curve analysis.

**References**

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[2] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[3] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy

optimization. In International Conference on Machine Learning (pp. 1889-1897).

[4] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., … & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484-489.

[5] Lillicrap, T. P., et al. (2015). "Continuous Control with Deep Reinforcement Learning." *arXiv preprint arXiv:1509.02971*.

[6] Kober, J., Bagnell, J. A., & Peters, J. (2013). "Reinforcement Learning in Robotics: A Survey." *International Journal of Robotics Research*.