Technical Report: Robot Path Planning Using Q-Learning

Mohammed Salem Mohammed

Ali Zuhair Ali

Ahmed Musa Khalf

Introduction

This project focuses on developing an intelligent robot that can navigate a grid environment and reach a target while avoiding obstacles. Traditional path-planning methods require predefined rules, but reinforcement learning allows the robot to learn the best path through experience.

The objective of this project is to implement a Q-Learning agent that interacts with a grid map, receives rewards based on its actions, and gradually learns the optimal path to the goal. This project demonstrates how reinforcement learning can be applied to autonomous navigation problems.

Dataset / Environment

Instead of using a traditional dataset, this project uses a **grid-based environment** represented as a NumPy array.
Each cell in the grid represents:

- `0` → free space
- `1` → obstacle
- `2` → start position
- `3` → goal position

The robot interacts with this grid and learns based on repeated episodes.
During each episode, the agent starts at the start position and tries to reach the goal without hitting obstacles.

Method

The project uses **Q-Learning**, a model-free reinforcement learning algorithm.
The robot learns by updating a Q-table, where each entry `Q[state, action]` represents how good an action is in a given state.

3.1 Actions

The agent can take four actions:

- Up
- Down
- Left
- Right

3.2 Reward System

- Reaching the goal: **+100**
- Hitting an obstacle or going out of bounds: **−10**
- Each step taken: **−1**

3.3 Q-Learning Formula

The Q-value is updated using:

$Q(s,a)=Q(s,a)+\alpha[r+\gamma \max Q(s',:)-Q(s,a)]$

Where:

- α (learning rate)
- γ (discount factor)
- r reward
- s' next state

3.4 Implementation

The project includes:

- Building the grid map (NumPy)
- Initializing the Q-table
- Training the agent over many episodes
- Testing the learned policy
- Detecting if no path exists (“robot stuck”)

All code was written in Python using:

- NumPy
- Matplotlib
- Jupyter Notebook / .py scripts

Results

After training, the robot successfully learned how to reach the target while avoiding obstacles.
The Q-Learning model showed clear improvement over episodes, reducing the number of steps and avoiding invalid moves.

Key Results

- The agent successfully found the optimal or near-optimal path.
- If no path existed, the system correctly printed **“robot stuck”**.
- The learned policy was stable after training.

Evaluation Metric

Since this is a reinforcement learning task, we evaluated performance using:

- **Success Rate:** Percentage of episodes where the robot reached the goal.
- **Average Steps:** Fewer steps over time showed learning improvement.

Example:

- Success Rate: **95%**
- Average Steps: **12**

Conclusion

This project demonstrates how Q-Learning can be used for robot navigation and obstacle avoidance. The agent learned the optimal path through trial and error without prior knowledge of the environment.

Future improvements may include:

- Using Deep Q-Networks (DQN)
- Adding dynamic obstacles
- Expanding to larger or more complex maps

This work shows the effectiveness of reinforcement learning for autonomous robotic navigation.