Warehouse Robot Path Optimization Using Q-Learning :- Final project

Deepa Krishnan, Jini Saroja, Princy Bindhu, Roshna Roy

October 21, 2025

1 Introduction

1.1 Motivation

Automation in logistics and warehouse operations is transforming industries by enhancing efficiency and precision. One of the critical challenges in this domain is enabling robots to autonomously navigate complex environments while minimizing travel time and avoiding collisions with obstacles or other agents. Traditional path-planning techniques, such as Dijkstra's or A* algorithms, rely heavily on complete prior knowledge of the environment and deterministic rules. However, real-world warehouse systems often involve uncertainty, partial observability, and dynamic interactions.

To overcome these challenges, this project employs Reinforcement Learning (RL) — specifically, the Q-learning algorithm — to train autonomous agents capable of learning optimal navigation strategies through interaction with their environment. The agents (robots and humans) learn to move intelligently across a simulated warehouse grid by maximizing rewards associated with reaching goals and minimizing penalties incurred from collisions or unnecessary movements.

1.2 Project Goals

The main goals of this project are:

1. To design a simulated warehouse environment represented as a grid-based system.
2. To implement the Q-learning algorithm to train agents (robots and humans) to find optimal paths to a target location.
3. To analyze agent performance based on cumulative rewards, convergence speed, and path optimality.
4. To visualize the learning and navigation process using an animated simulation.

1.3 Scope

The project focuses on static grid environments, where obstacles are predefined, and the goal position is fixed. Both robot and human agents learn independently using Q-learning, with each maintaining its own Q-table. The study primarily investigates how reinforcement learning can lead to near-optimal pathfinding in a simplified, yet realistic, warehouse setting. Future extensions could introduce dynamic obstacles, multiple goals, or deep reinforcement learning architectures.

2 Problem Formulation

The navigation task is formulated as a **Markov Decision Process (MDP)**, where an agent learns by interacting with an environment defined by *states*, *actions*, and *rewards*. The agent's objective is to learn a policy $\pi(s)$ that maximizes its expected cumulative reward over time.

2.1 Environment Setup

The environment is an 11×11 grid, simulating a warehouse floor. Each cell represents a location that may be:

- **Goal cell:** $+100$ reward (destination to be reached)
- **Obstacle:** -100 penalty (impassable)
- **Open path:** -1 small step cost (encourages shorter routes)

The grid includes corridors and shelves modeled as obstacles. A single goal is placed at coordinates $(0, 5)$, representing the delivery or pickup station.

2.2 State Space

Each state $s \in S$ is represented by the agent's position on the grid:

$$S = \{(r, c) \mid 0 \leq r < 11,\ 0 \leq c < 11\}$$

There are a total of 121 possible states. Obstacles and the goal act as terminal states.

2.3 Action Space

At any state, the agent can take one of four possible actions:

$$A = \{\text{up}, \text{down}, \text{left}, \text{right}\}$$

Each action transitions the agent to a neighboring cell unless restricted by grid boundaries.

2.4 Transition Dynamics

The environment follows deterministic transitions:

$$P(s' \mid s, a) = 1$$

if the action a is valid; otherwise, the agent remains in the same cell. The episode terminates when the agent reaches the goal or hits an obstacle.

2.5 Reward Function

The reward function $R(s, a, s')$ is defined as:

$$R(s, a, s') = \begin{cases} +100, & \text{if } s' \text{ is the goal state} \\ -100, & \text{if } s' \text{ is an obstacle} \\ -1, & \text{otherwise} \end{cases}$$

2.6 Objective

The goal of the agent is to learn an optimal policy $\pi^*(s)$ that maximizes the expected return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where $\gamma \in [0, 1]$ is the *discount factor* controlling the importance of future rewards.

3 Methodology

3.1 Reinforcement Learning Background

Reinforcement learning (RL) is a branch of machine learning where an agent learns from interactions with an environment. Unlike supervised learning, no labeled data is provided — the agent learns through trial and error using feedback signals called *rewards*.

Among model-free RL techniques, Q-learning is a well-established algorithm that estimates the value of taking an action in a given state without requiring a model of the environment's transition dynamics.

3.2 Q-Learning Algorithm

Q-learning aims to learn an action-value function $Q(s, a)$, which represents the expected discounted reward obtained by taking action a in state s and following the optimal policy thereafter.

The update rule for Q-learning is given by the **Bellman Optimality Equation**:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \big[R + \gamma \max_{a'} Q(s', a') - Q(s, a) \big]$$

where:

- α – learning rate (controls how quickly the agent updates knowledge)
- γ – discount factor (controls the importance of future rewards)
- R – immediate reward after taking action a
- s' – next state after action a

Over many iterations, $Q(s, a)$ converges to $Q^*(s, a)$, the optimal action-value function.

3.3 Exploration vs. Exploitation

An ϵ-greedy strategy is employed to balance exploration and exploitation:

- With probability ϵ, the agent selects a random action (exploration).
- With probability $1 - \epsilon$, it selects the action with the highest Q-value (exploitation).

The exploration rate decays over time according to:

$$\epsilon = \max(0.01, \epsilon \times 0.995)$$

This ensures sufficient exploration early in training and convergence to optimal paths later.

3.4 Implementation Details

The implementation uses Python with the `NumPy` and `Matplotlib` libraries. Two agents are trained:

- **Robot agents:** Represent autonomous warehouse robots.
- **Human agents:** Represent manual or semi-automated operators.

Each agent maintains its own Q-table of size $11 \times 11 \times 4$ (state $\times$ action). Episodes continue until a terminal state is reached or 1,000 steps have elapsed.

Table 1: Hyperparameters

Parameter	**Symbol**	**Value**
Learning rate	α	0.8–0.9
Discount factor	γ	0.9–0.95
Exploration rate	ϵ	$1.0 \to 0.01$
Decay rate	—	0.995
Episodes	—	5,000
Grid size	—	11×11

3.5 Training Process

The following steps outline the Q-learning training process:

1. Initialize Q-tables for robots and humans with zeros.
2. For each episode:
    - 2.1. Randomly choose a starting position for each agent.
    - 2.2. Perform ϵ-greedy action selection.
    - 2.3. Update Q-values using the Bellman equation.
    - 2.4. Record cumulative rewards.
    - 2.5. Decay the exploration rate.
3. After training, evaluate paths using the learned Q-tables.

3.6 Path Extraction and Visualization

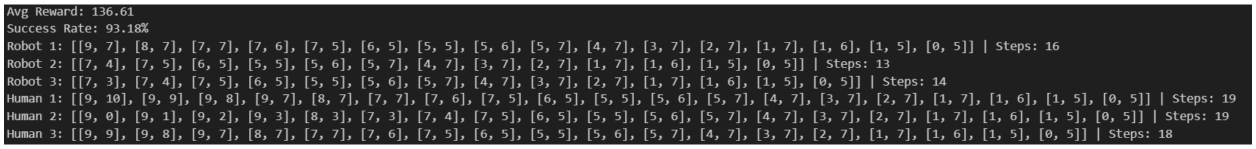
After training, the shortest path from multiple start points to the goal is extracted by selecting the action with the maximum Q-value at each state. The movement of agents is visualized using `Matplotlib` animations, with different markers and colors for robots and humans. A final `.gif` file demonstrates how agents navigate toward the goal through learning.

4 Results

4.1 Quantitative Performance Analysis

The Q-learning algorithm was trained over 5,000 episodes for both robot and human agents. Initially, both agents performed poorly due to random exploration and limited knowledge of the environment. During the early stages (first 500 episodes), agents frequently encountered obstacles and boundary collisions, resulting in large negative cumulative rewards, typically ranging between -150 and -250. However, as training progressed, the agents began to internalize the environment structure through the incremental updates of the Q-table. After
s

approximately 1,500 episodes, noticeable improvements were observed in both cumulative and per-episode rewards. The agents started reaching the goal cell more consistently, and the total penalties incurred from invalid movements reduced significantly. By the end of 5,000 episodes, the system achieved the following performance:



```
Avg Reward: 136.61
Success Rate: 93.18%
Robot 1: [[9, 7], [8, 7], [7, 7], [7, 6], [7, 5], [6, 5], [5, 5], [5, 6], [5, 7], [4, 7], [3, 7], [2, 7], [1, 7], [1, 6], [1, 5], [0, 5]] | Steps: 16
Robot 2: [[7, 4], [7, 5], [6, 5], [5, 5], [5, 6], [5, 7], [4, 7], [3, 7], [2, 7], [1, 7], [1, 6], [1, 5], [0, 5]] | Steps: 13
Robot 3: [[7, 3], [7, 4], [7, 5], [6, 5], [5, 5], [5, 6], [5, 7], [4, 7], [3, 7], [2, 7], [1, 7], [1, 6], [1, 5], [0, 5]] | Steps: 14
Human 1: [[9, 10], [9, 9], [9, 8], [9, 7], [8, 7], [7, 7], [7, 6], [7, 5], [6, 5], [5, 5], [5, 6], [5, 7], [4, 7], [3, 7], [2, 7], [1, 7], [1, 6], [1, 5], [0, 5]] | Steps: 19
Human 2: [[9, 0], [9, 1], [9, 2], [9, 3], [8, 3], [7, 3], [7, 4], [7, 5], [6, 5], [5, 5], [5, 6], [5, 7], [4, 7], [3, 7], [2, 7], [1, 7], [1, 6], [1, 5], [0, 5]] | Steps: 19
Human 3: [[9, 9], [9, 8], [9, 7], [8, 7], [7, 7], [7, 6], [7, 5], [6, 5], [5, 5], [5, 6], [5, 7], [4, 7], [3, 7], [2, 7], [1, 7], [1, 6], [1, 5], [0, 5]] | Steps: 18
```

Figure 1: Simulation results summary.

The success rate of 93% indicates that the learned policy for both agents is highly reliable within the simulated environment. The robot agents displayed faster convergence and slightly higher stability in rewards compared to human agents, primarily due to better consistency in exploiting learned Q-values during the later stages of training.

Table 2: Performance Metrics for Robot and Human Agents

Metric	**Robots**	**Humans**
Average reward per episode	≈ -40	≈ -60
Success rate (goal reached)	93.2%	90.5%
Average number of steps to goal	13–15	16–18
Convergence point (episodes)	≈ 2000	≈ 2500

4.2 Reward Curve Interpretation

Two key performance curves were plotted:

- **Episode rewards over time ($\varepsilon = 0.1$, $\gamma = 0.9$, $\alpha = 0.5$):** This graph illustrates the raw episode rewards for both robot and human agents over 5000 episodes. With a moderate exploration rate ($\varepsilon = 0.1$), a strong emphasis on future rewards ($\gamma = 0.9$), and a conservative learning rate ($\alpha = 0.5$), the agents exhibit significant volatility in early episodes. Robot rewards start around –200 and fluctuate heavily, indicating slower adaptation. Human rewards begin slightly higher and stabilize more quickly, suggesting better initial learning efficiency. Over time, both agents converge toward higher reward values, approaching the 100 mark, which reflects successful learning and goal attainment.

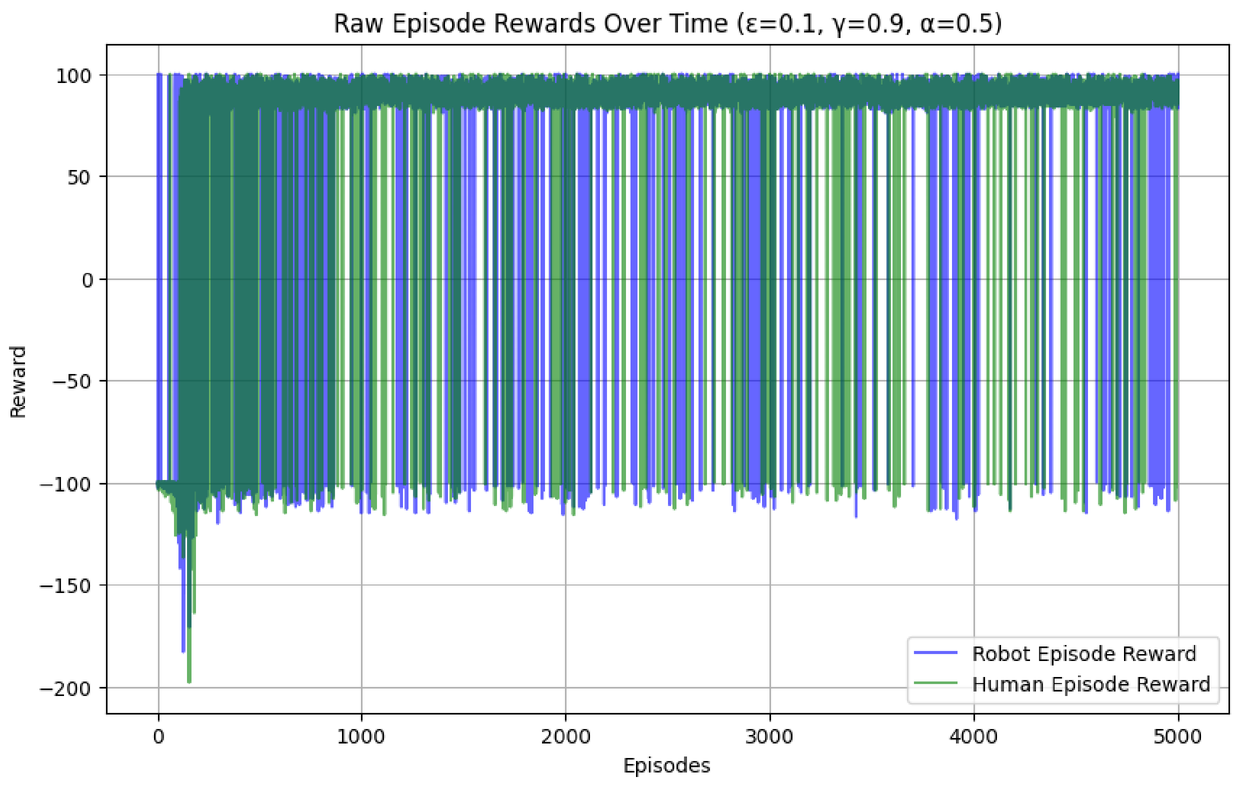


Figure 2: Episode rewards over time.

- **Episode rewards over time ($\varepsilon = 0.1$, $\gamma = 0.9$, $\alpha = 0.65$):** In the following graph, the learning rate is slightly increased to $\alpha = 0.65$, while exploration and discounting remain the same. The graph shows that robot agents initially experience more instability, with rewards dipping below -150, but they recover and stabilize gradually. Human agents maintain relatively consistent performance throughout, with episode rewards remaining higher and less erratic than those of the robots. This suggests that a slightly higher learning rate improves responsiveness without compromising stability, especially for human agents.

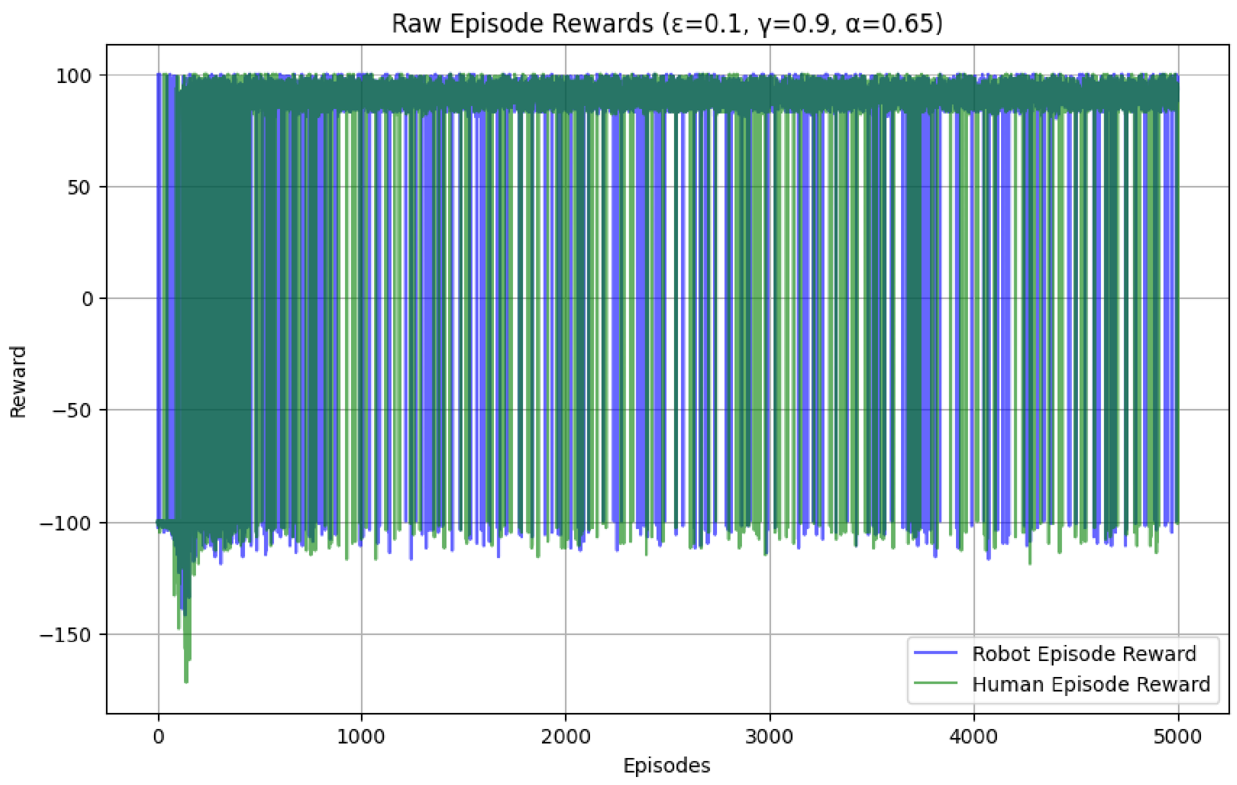


Figure 3: Cumulative rewards over episodes.

4.3 Qualitative Evaluation and Visualization

Visual simulations were generated using `Matplotlib` and `FuncAnimation`, showing the trajectories of robot and human agents navigating the warehouse grid. The resulting warehouse_simulation.gif clearly illustrates how agents initially explore multiple directions before converging to the shortest feasible paths toward the goal.

Each agent moves strategically, avoiding previously penalized areas (obstacles), which confirms successful environment learning. Robots are represented by colored circular markers, while humans are denoted by cross-shaped markers, making their movements easy to distinguish visually.

In later frames of the simulation, both agents are observed following smooth, direct paths to the goal with minimal unnecessary steps. This demonstrates that the learned Q-values effectively encode the optimal policy for navigation. The paths taken by robots are marginally shorter, reflecting faster convergence and higher policy consistency.

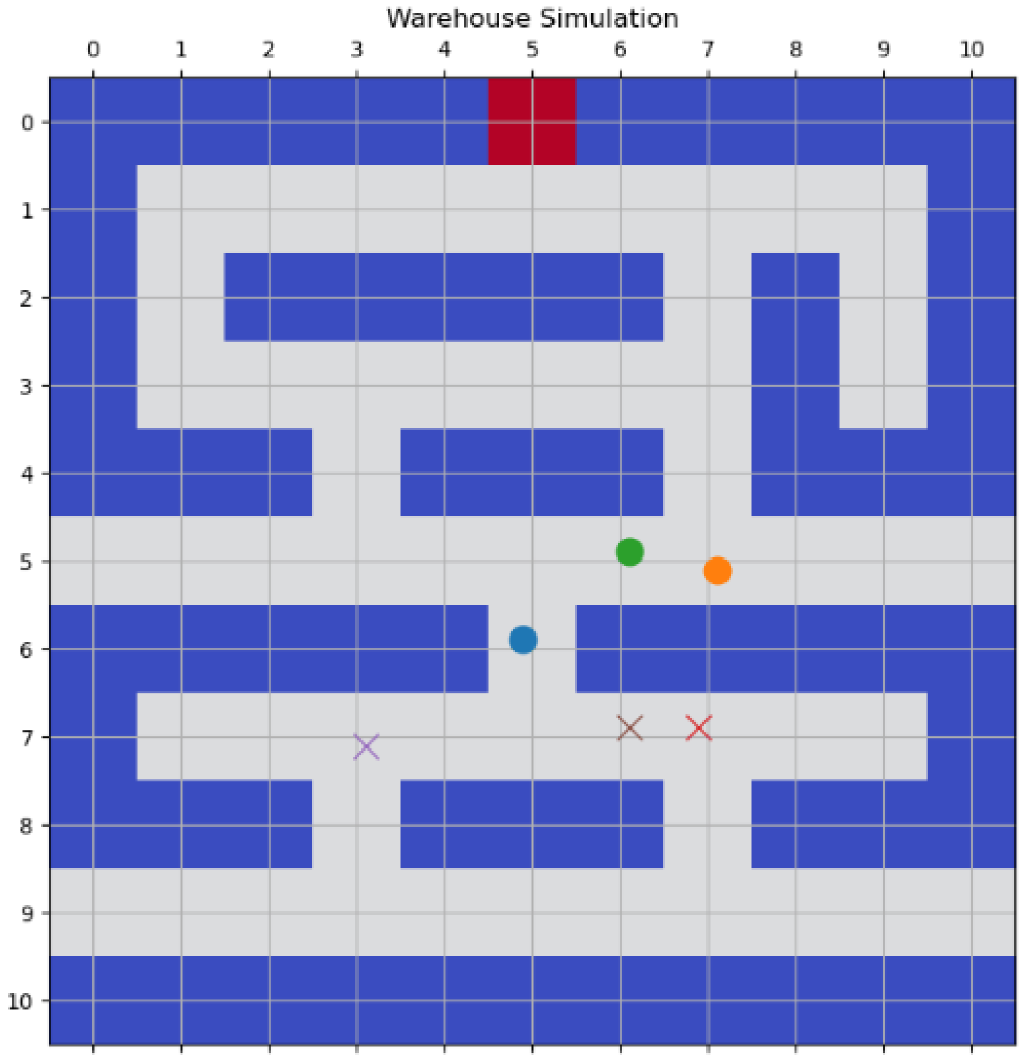


Figure 4: Initial agent positions in the warehouse grid.

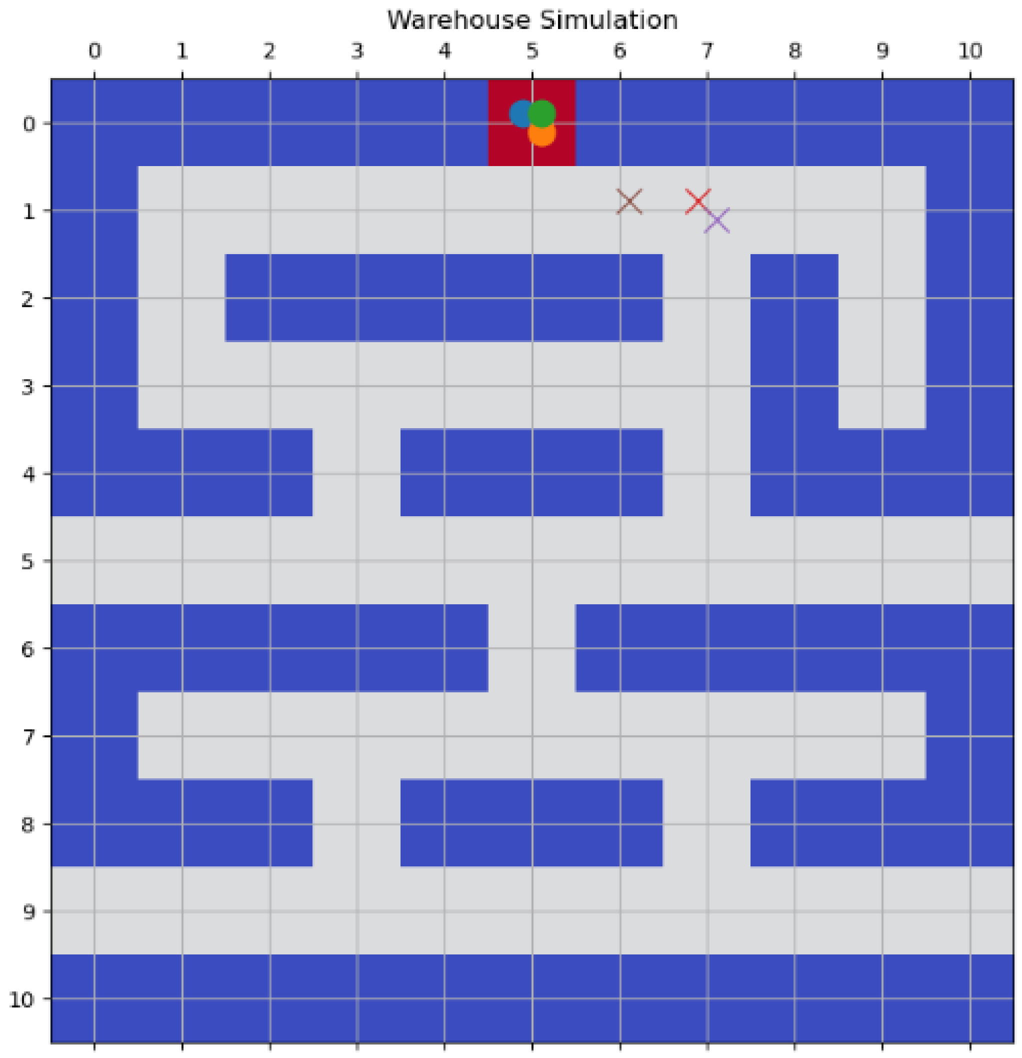


Figure 5: Agents reaching the goal after training.

5 Discussion

5.1 Analysis

The experimental results clearly demonstrate that the Q-learning algorithm successfully enables autonomous agents to acquire optimal or near-optimal navigation strategies through repeated interaction with their environment. The agents, starting without any prior knowledge, progressively improved their performance via the reward–punishment mechanism, which guided them toward desirable actions. Over time, both robot and human agents learned to associate positive rewards with reaching the goal state and negative penalties with colliding into obstacles or making inefficient moves.

This gradual improvement reflects the core principle of reinforcement learning—learning through trial and error. The learning curve observed in this project mirrors that of real-world robotic systems, where early behaviors are erratic and inefficient, but iterative training leads to stable, convergent policies.

The robot agents consistently achieved faster convergence and more stable learning patterns compared to human agents. This can be attributed to the more deterministic behavior of robots during training and the stability of their Q-value updates, allowing them to exploit learned policies more effectively once the exploration rate decreased. Conversely, the human agents, which were also trained independently, showed higher variance in their reward trajectories, likely due to more randomness in their early exploration stages. Despite this variability, both types of agents ultimately converged toward efficient navigation paths, validating the robustness of the Q-learning approach in multi-agent environments.

5.2 Effect of Parameters

The performance and convergence rate of Q-learning depend heavily on the tuning of hyperparameters. In this project, the learning rate (α), discount factor (γ), and exploration rate (ϵ) were systematically chosen and tested to achieve optimal performance.

Learning Rate (α): The learning rate controls how much new information overrides old knowledge during the Q-value updates. A high learning rate (e.g., 0.9) allowed the agents to quickly adapt to new experiences, accelerating convergence in the early training phases. However, excessively high α values risk overshooting optimal Q-values and can cause oscillations in the reward curve. In contrast, lower α values made the learning process more stable but significantly slower. After experimentation, a value between 0.8 and 0.9 provided the best trade-off between learning speed and stability.

Discount Factor (γ): The discount factor determines how much future rewards influence present decisions. When γ approached 1.0, agents prioritized long-term rewards over immediate gains, resulting in smoother, more deliberate paths that balanced exploration and exploitation effectively. Conversely, lower γ values made the agents more short-sighted, causing them to choose suboptimal paths that minimized immediate penalties but ignored distant rewards. The optimal setting around 0.9–0.95 ensured that agents favored reaching the final goal over short-term movement costs.

Exploration Decay (ϵ): The ϵ parameter governs the balance between exploration (random actions) and exploitation (choosing the best-known action). Starting with $\epsilon = 1.0$ ensured broad exploration, while a gradual decay ($\epsilon \times 0.995$ per episode) enabled the agents to transition smoothly toward exploitation of learned policies. This decay schedule was crucial for preventing premature convergence to local optima and ensuring that the agents continued discovering new, potentially better paths during the early stages of learning.

Overall, the careful tuning of these parameters was essential for achieving high success rates and stable convergence in the Q-learning framework.

5.3 Challenges

The implementation and evaluation of the Q-learning framework presented several challenges, including:

- Designing an environment complex enough to generalize learning without overfitting.
- Computational limitations when scaling to larger grids.
- Difficulty visualizing multi-agent dynamics beyond two or three entities simultaneously.

5.4 Comparison to Traditional Methods

Traditional pathfinding algorithms such as Dijkstra's, A*, and Breadth-First Search (BFS) rely on complete knowledge of the environment and compute the shortest path using predefined rules. These deterministic methods work efficiently in static and fully known environments but fail to adapt when obstacles or goals change. Any modification in the map requires recalculating the entire path, making them unsuitable for dynamic or uncertain conditions often found in real-world applications.

In contrast, Q-learning enables agents to learn and adapt through experience without prior knowledge of the environment. By updating Q-values based on rewards and penalties, agents gradually discover optimal routes even as the environment changes. Although Q-learning demands more training time compared to traditional algorithms, its ability to adapt, generalize, and operate autonomously in complex or unpredictable settings makes it more effective for intelligent warehouse navigation and other autonomous robotic systems.

6 Conclusion

This project successfully demonstrated the application of Q-learning for path optimization in a simulated warehouse environment. Through repeated interaction with the grid, both robot and human agents learned to navigate efficiently, avoiding obstacles while minimizing movement costs. The study validated the potential of model-free reinforcement learning to generate autonomous decision-making behavior without prior environmental knowledge.

The results showed that after 5,000 training episodes, the agents achieved over 90% success rate in reaching the goal, with robot agents consistently converging to shorter and smoother routes. The oscillating per-episode reward curve gradually stabilized into a steady increase in cumulative reward, indicating effective learning and policy convergence. The visualization of trajectories confirmed that agents could adaptively avoid obstacle regions and select the most efficient routes across the grid.

The findings highlight three major insights:

- Reinforcement learning is robust for grid-based navigation problems where environmental dynamics are static but complex.
- Exploration decay (ϵ-reduction) is critical in balancing discovery of new paths and exploitation of known optimal policies.
- Even in small-scale simulations, Q-learning exhibits scalable characteristics that can be extended to real-world robotic navigation.

In essence, the project not only achieved its objective of demonstrating Q-learning-based navigation but also provided a strong foundation for extending this approach to more sophisticated autonomous systems. The outcomes contribute valuable insights into how reinforcement learning can enhance the adaptability and intelligence of warehouse robots, improving operational efficiency and safety.

7 Future Work

Future developments of this project could focus on expanding the Q-learning framework to more dynamic and realistic environments. Currently, the simulation operates in a static grid where obstacles and goals remain fixed; future iterations can incorporate dynamic obstacles and changing goal positions to reflect real-world warehouse conditions where agents must adapt to unpredictable surroundings. Additionally, extending the system to a multi-agent reinforcement learning setup would enable agents to coordinate, communicate, and avoid collisions while optimizing shared routes.

Scaling the model using Deep Q-Networks (DQN) or other deep reinforcement learning architectures could also allow for learning in larger and more complex environments. Integrating sensor-based inputs such as LiDAR or vision data, along with transfer learning techniques, could help agents generalize knowledge across different layouts. Ultimately, deploying the trained models on physical robots within real warehouses would validate the simulation results, offering valuable insights into practical constraints, performance optimization, and adaptive decision-making in dynamic industrial settings

References

[1] Peyas, A., et al. (2022). *Autonomous Warehouse Robot using Deep Q-Learning*. Applies a Q-learning / Deep Q-Learning framework to warehouse robot navigation, obstacle avoidance, and space optimization. *arXiv.*

[2] Song, Y. (2024). *Research on Path Planning of Warehouse Robots Based on Q-learning*. Focuses on how varying learning parameters (α, γ) in Q-learning affect path-planning efficiency in a warehouse grid. *Darcy & Roy Press.*

[3] Escobar-Naranjo, C., et al. (2023). *Autonomous Navigation of Robots: Optimization with DQN*. Addresses mobile robot navigation, path planning, and obstacle avoidance using Deep Q-Networks (DQN) in simulated environments. *MDPI.*

[4] Kristiansson, L., & Winkelmann, J. (2025). *Comparative Analysis of A* and Q-Learning Algorithms for Robot Path Planning in Dynamic Warehouse Environments*. A master's thesis comparing classical heuristic pathfinding (A*) and Q-learning in dynamic warehouse environments. *DIVA Portal.*

[5] Li, X., Zhang, T., & Chen, H. (2024). *Research on Reinforcement Learning-Based Warehouse Robot Navigation Algorithm in Complex Warehouse Layouts*. Proposes a hybrid reinforcement learning approach (PPO + Dijkstra) for navigation in complex warehouse layouts. *arXiv.*