UNIVERSITY OF MASSACHUSETTS DARTMOUTH
DEPARTMENT OF COMPUTER & INFORMATION SCIENCE

Reinforcement Learning for Intelligent Room Mapping and Navigation

CIS 600: MASTER'S PROJECT
FINAL REPORT

PRESENTED BY
Shuayib Abdulkadir (01801970)

PROJECT ADVISOR
Professor Amir AkhavanMasoumi

Date: December 2024

Introduction

Robotic vacuum cleaners are a cornerstone of modern smart homes, blending autonomous navigation with adaptive cleaning capabilities. This project explores the development and training of a robot vacuum cleaner using a machine learning approach, specifically Deep Q-Networks (DQN), to navigate and clean rooms efficiently. The aim is to train the robot to learn optimal cleaning strategies while avoiding obstacles and minimizing redundant movements.

Motivation

Traditional robotic vacuums rely on pre-programmed patterns or basic heuristics for cleaning. While functional, these methods often result in inefficiencies, such as repeated cleaning of the same area or missing spots entirely. This project aims to overcome these limitations by leveraging reinforcement learning to enable the vacuum cleaner to learn from its environment and adapt dynamically to new layouts and challenges.

Objective

The primary objective of this project is to design and implement a simulation environment where a robot vacuum cleaner, trained using reinforcement learning, can:

1. Develop a Simulation Environment: Create a grid-based environment with dirt, obstacles, and furniture.
2. Implement RL Algorithms: Use Q-Learning and DQN to train the robot.
3. Optimize Cleaning Strategies: Minimize redundant movements and collisions while maximizing coverage.
4. Real-Time Visualization: Display the robot's performance and decision-making process.

Scope

The scope of this project includes:

- Developing a grid-based simulation environment to represent rooms with dirt, obstacles, and furniture.
- Implementing a DQN algorithm to train the robotic vacuum cleaner.
- Visualizing the robot’s movements and performance metrics in real-time.
- Generating statistical insights about the robot’s learning process and efficiency.
- Providing a foundation for future work in real-world robotic vacuum applications.

Literature Review

Several studies have explored the application of reinforcement learning in robotics, particularly for navigation and task optimization. DQN has been widely adopted due to its ability to handle discrete action spaces and efficiently approximate Q-values. In the domain of robotic vacuums, prior work has largely focused on heuristic-based navigation or SLAM (Simultaneous Localization and Mapping) techniques. This project extends these efforts by applying DQN for task-specific optimization in cleaning scenarios.

System Analysis and Functional Requirements

- The system should simulate grid-based room layouts with configurable obstacles and dirt tiles.
- The robotic vacuum should autonomously navigate the environment and clean dirt tiles.
- A backend server should store simulation results, and a frontend should visualize the robot's performance.

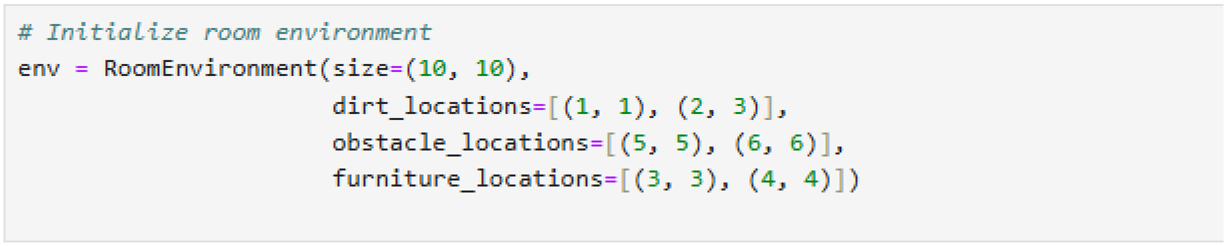
Non-Functional Requirements

- The simulation should run efficiently on standard hardware.
- The system should provide real-time visual feedback.
- The architecture should support future enhancements, such as integrating IoT sensors or transitioning to continuous-space navigation.

Requirement Analysis

The project requires:

1. A simulation environment for grid-based navigation.
2. A DQN implementation for reinforcement learning.
3. Visualization tools for real-time rendering and performance tracking.
4. A backend for storing and retrieving simulation data.
5. Libraries like TensorFlow, Pygame, Flask, and MongoDB.



```
# Initialize room environment
env = RoomEnvironment(size=(10, 10),
                      dirt_locations=[(1, 1), (2, 3)],
                      obstacle_locations=[(5, 5), (6, 6)],
                      furniture_locations=[(3, 3), (4, 4)])
```

Fig 1: Initialize room environment

Implementation and Results

Reinforcement Learning with Q-Learning and Deep Q-Network (DQN)

Q-Learning Algorithm

Q-Learning is a model-free reinforcement learning (RL) algorithm designed for discrete state-action spaces, making it ideal for environments such as our robot vacuum cleaner simulation.

The algorithm learns optimal action policies by iteratively updating Q-values based on rewards and penalties received during interaction with the environment. It achieves this without requiring a model of the environment's dynamics, which enhances its adaptability to various scenarios.

The Q-value represents the expected utility of taking a specific action in each state, enabling the agent to make decisions that maximize long-term rewards.

Key Features:

- **Model-Free:** Does not require prior knowledge of the environment.
- **Optimal Policy:** Learns the best action to take in any given state by iteratively updating Q-values.

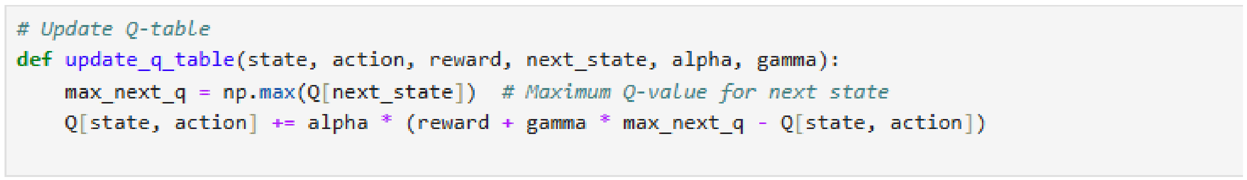
Q-Learning Update Rule:

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha\big(R(s,a) + \gamma \max_{a'} Q(s',a')\big)$$

Where:

- $Q_t(s,a)$: Current Q-value estimate.
- α: Learning rate.
- $R(s,a)$: Reward for taking action a in state s.
- γ: Discount factor for future rewards.

Update Rule Implementation:



```
# Update Q-table
def update_q_table(state, action, reward, next_state, alpha, gamma):
    max_next_q = np.max(Q[next_state])  # Maximum Q-value for next state
    Q[state, action] += alpha * (reward + gamma * max_next_q - Q[state, action])
```

Fig 2: update rule implementation

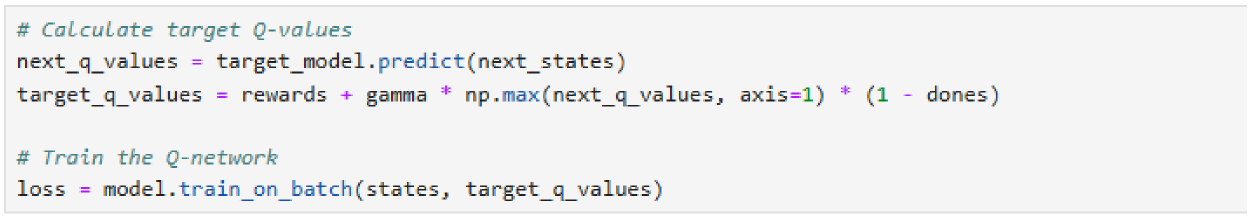
DQN Implementation

For environments with larger state spaces, DQN approximates Q-values using neural networks.

Key Features:

1. Neural Network: Predicts Q-values for given states.
2. Experience Replay: Stores past experiences to break data correlation.
3. Target Network: Provides stable training targets.

DQN Update Logic:



```
# Calculate target Q-values
next_q_values = target_model.predict(next_states)
target_q_values = rewards + gamma * np.max(next_q_values, axis=1) * (1 - dones)

# Train the Q-network
loss = model.train_on_batch(states, target_q_values)
```

Fig 3: DQN update table

Reward System

The reward function incentivizes efficient cleaning:

- Positive Rewards: +10 for cleaning dirt tile.
- Negative Rewards:
    - −5 for hitting an obstacle or furniture.

    - −1 for unnecessary steps.
- Completion Bonus: +50 for cleaning all dirty tiles.

Performance Improvement:

- Early episodes show negative rewards due to collisions and inefficiency.
- After ~100 episodes, the agent learns to clean efficiently, achieving high positive rewards.
- Stabilized learning curve indicates successful policy convergence.

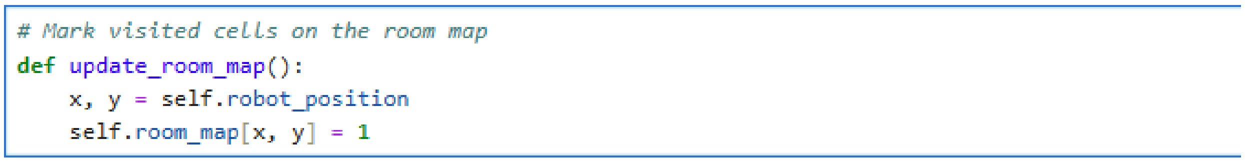
Impact of DQN:

- Combines Q-Learning principles with neural networks for complex state spaces.
- Learns optimal cleaning strategies effectively.

Visualization and Results

1. **Real-Time Visualization:**
    - Pygame renders the robot's trajectory, obstacles, and cleaned tiles.
2. **Performance Metrics:**
    - Cumulative rewards per episode indicate learning progress.
    - Charts visualize trends in efficiency over episodes.
3. **Video Generation:**
    - Captured frames are converted into an MP4 video for analysis using FFmpeg.

Trajectory Mapping:



```
# Mark visited cells on the room map
def update_room_map():
    x, y = self.robot_position
    self.room_map[x, y] = 1
```

Fig 4: trajectory Mapping

Results Analysis and Performance Trends

- Early Episodes: Show negative rewards due to collisions and inefficient exploration.
- Learning Phase: The agent learns to avoid obstacles and optimize its trajectory, achieving higher rewards.

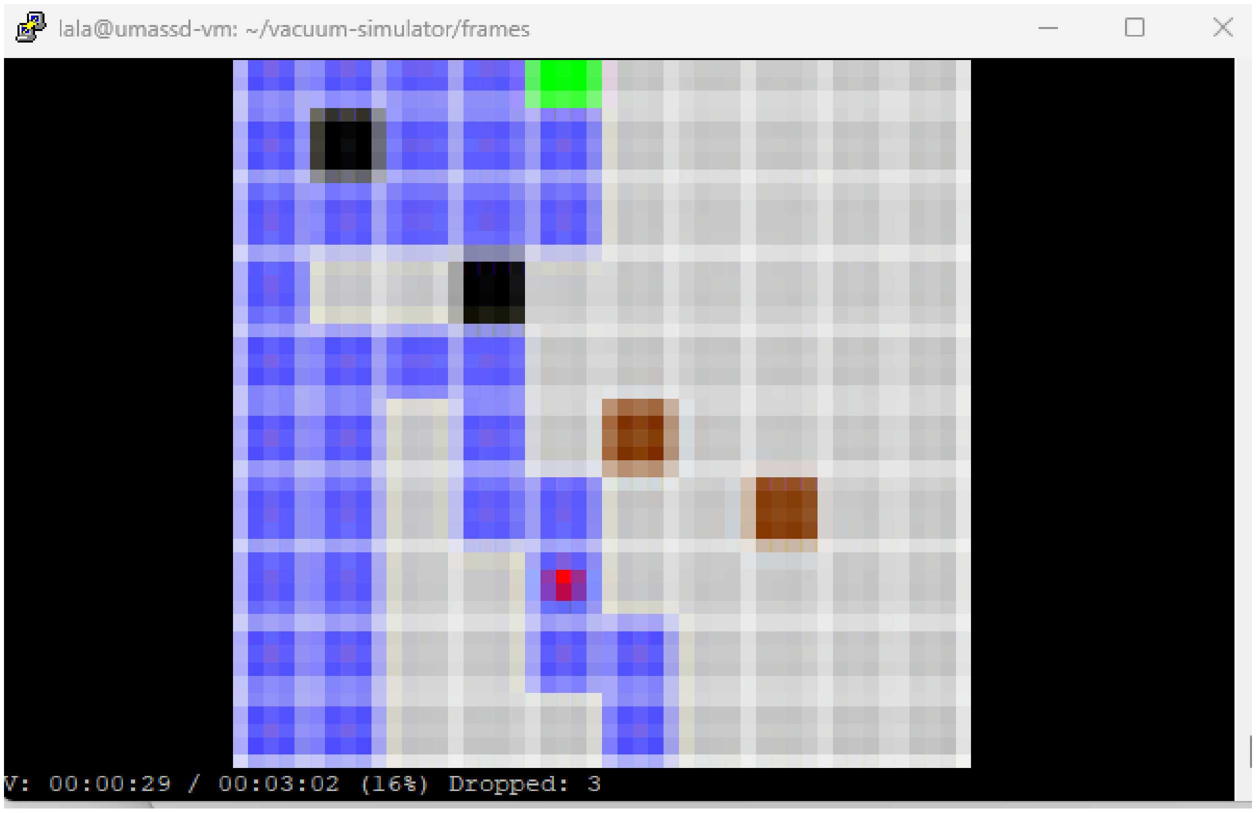
- Stabilization: After ~100 episodes, the learning curve stabilizes, indicating convergence.

Efficiency Metrics

Episode	Total Reward
1	-500
50	200
100	500

table 1: efficiency metrics

Visualization of Agent trajectory and navigation in action



```
V: 00:00:29 / 00:03:02 (16%) Dropped: 3
```

Fig 5: Vacuum Simulator Trajectory and Navigation

Conclusion and Future Enhancements

Conclusion

This project demonstrates the effective application of reinforcement learning in autonomous robotic systems, particularly in the context of room mapping and navigation for robotic vacuum cleaners. By utilizing both Q-Learning and Deep Q-Networks (DQN), the robot learned to navigate efficiently, avoid obstacles, and clean dirt tiles in a simulated grid-based environment.

Key takeaways include:

1. Scalability: Q-Learning provided a solid foundation for learning optimal cleaning strategies in smaller environments, while DQN extended this scalability to handle larger, more complex state spaces.
2. Adaptability: The robot adapted dynamically to various room layouts, showcasing the versatility of reinforcement learning algorithms.
3. Efficiency: Over the training episodes, the agent demonstrated significant performance improvements, reducing redundant movements and collisions.
4. Visualization: Real-time simulation and visualization provided intuitive insights into the robot's decision-making and learning processes.
5. Practical Application: The combination of advanced algorithms and intuitive visualization forms a robust framework that can be extended to real-world scenarios, such as IoT-connected cleaning systems.

Overall, this project highlights how reinforcement learning, combined with effective visualization techniques, can address real-world challenges in robotics, contributing to the advancement of smart home technologies.

Future Enhancements

While the project successfully achieved its goals, there is significant potential for further development and improvements:

1. Advanced Algorithms:
   - Explore state-of-the-art reinforcement learning techniques such as Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), or Deep Deterministic Policy Gradient (DDPG) to handle continuous action spaces and more sophisticated scenarios.
2. Dynamic Obstacles:
   - Introduce moving obstacles to simulate real-world challenges where furniture or people may change positions during operation.

3. Continuous-Space Navigation:
    - Transition from grid-based navigation to continuous space representation for greater realism and applicability in real-world robotic systems.
4. IoT Integration:
    - Leverage IoT sensors to enable real-time data collection and interaction with physical robotic vacuum cleaners. This can enhance adaptability and performance in diverse environments.
5. Multi-Agent Collaboration:
    - Extend the framework to support multiple robotic agents working collaboratively to clean larger or multi-room environments.
6. Enhanced Reward Mechanisms:
    - Refine the reward system to account for energy efficiency, cleaning completeness, and time constraints, encouraging more realistic behaviors.
7. SLAM Integration:
    - Incorporate Simultaneous Localization and Mapping (SLAM) techniques to improve spatial awareness and navigation in unstructured and dynamic environments.
8. Real-World Deployment:
    - Develop a prototype to test the trained models on actual robotic vacuum hardware, bridging the gap between simulation and practical application.
9. Improved Visualization and Analytics:
    - Expand the visualization dashboard to include advanced analytics, heatmaps of visited areas, and time-lapse visualizations of the robot's performance over multiple episodes.
10. Multifunctionality:
    - Extend the system's functionality to include additional cleaning modes such as mopping or polishing, making it adaptable for a variety of cleaning tasks.

By pursuing these enhancements, the project can evolve into a comprehensive solution for intelligent robotic navigation and cleaning, paving the way for advanced applications in smart homes and autonomous systems

References

1. Donepudi, P. K., "Reinforcement Learning for Robotic Grasping and Manipulation: A Review," *Asia Pacific Journal of Energy and Environment*, vol. 7, no. 2, pp. 66-78, 2020.
2. Lu, H., Li, Y., Mu, S., Wang, D., Kim, H., and Serikawa, S., "Motor Anomaly Detection for Unmanned Aerial Vehicles Using Reinforcement Learning," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2315-2322, 2018.
3. Wang, H., Yuan, S., Guo, M., Chan, C.-Y., Li, X., and Lan, W., "Tactical Driving Decisions of Unmanned Ground Vehicles in Complex Highway Environments: A Deep Reinforcement Approach," *Journal of Automobile Engineering*, vol. 235, no. 4, pp. 1113-1127, 2020.
4. Vamvoudakis, K. G., Modares, H., Kiumarsi, B., and Lewis, F. L., "Game Theory-Based Control System Algorithms with Real-Time Reinforcement Learning: How to Solve Multiplayer Games Online," *IEEE Control Systems Magazine*, vol. 37, no. 1, pp. 33-52, 2017.
5. Kumar, R., Jitoko, P., Kumar, S., Pillay, K., Prakash, P., Sagar, A., Singh, R., and Mehta, U., "Maze Solving Robot with Automated Obstacle Avoidance," *Procedia Computer Science*, vol. 105, pp. 57-61, 2017.
6. Silaghi, H., Ecsedi, E., Mihok, E., and Spoiala, V., "The Development of an Autonomous Maze Robot," in *15th International Conference on Engineering of Modern Electric Systems*, Oradea, Romania, June 13-14, 2019.
7. Barnouti, N. H., Al-Dabbagh, S. S. M., and Naser, M. A. S. N., "Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm," *Journal of Computer and Communications*, vol. 4, no. 11, 2016.
8. Kathe, O., Turkar, V., Jagtap, A., and Gidaye, G., "Maze Solving Robot Using Image Processing," in *IEEE Bombay Section Symposium*, Mumbai, India, 2015.
9. Skinner, B. F., *Science and Human Behavior*, The Free Press; New Impression, 1953.
10. Sutton, S. R., and Barto, G. A., *Reinforcement Learning: An Introduction*, The MIT Press, 1998.