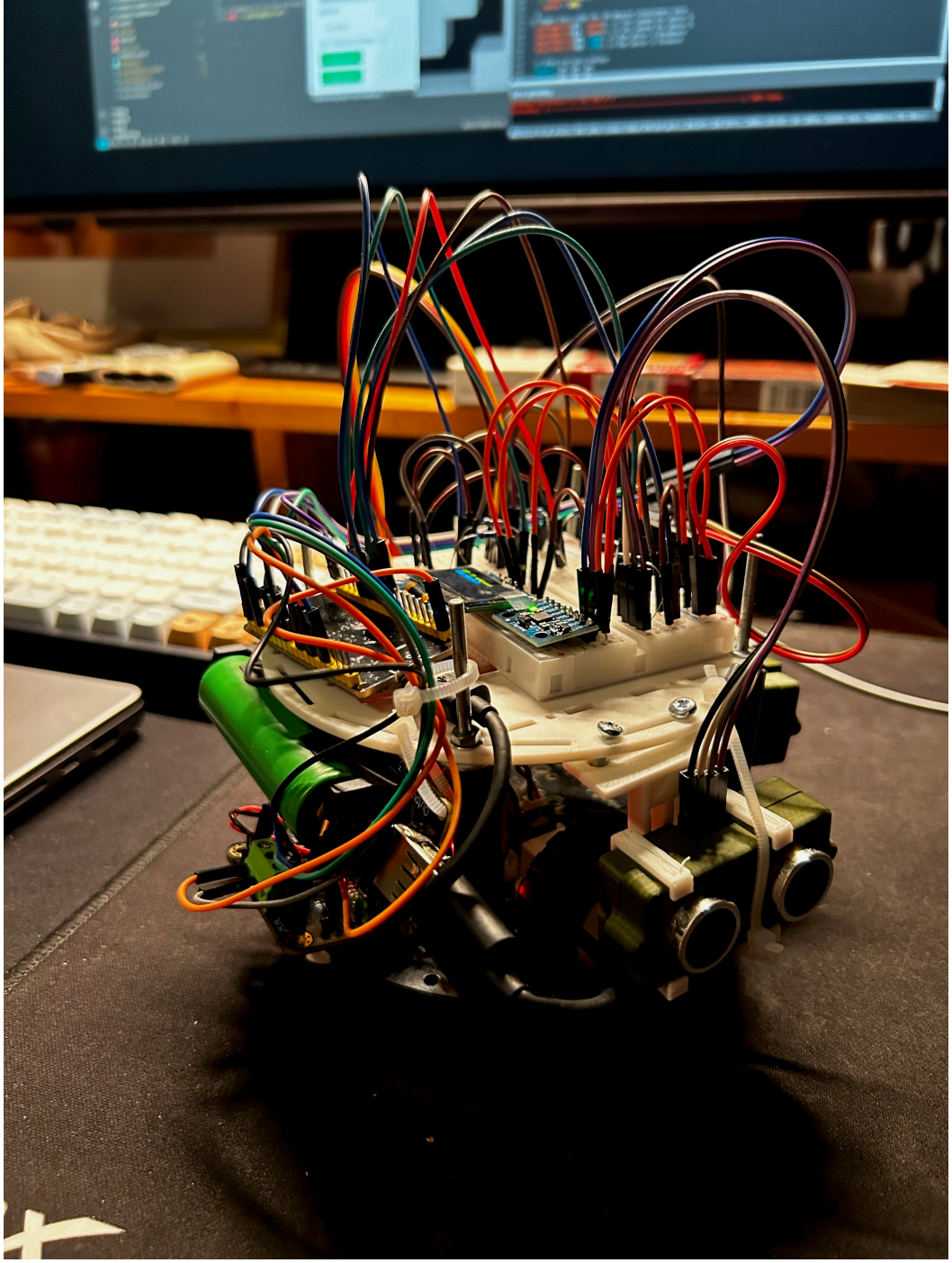


EXPLORING THE FEASIBILITY OF SIM2REAL TRANSFER IN REINFORCEMENT LEARNING

APPLICATION IN MAZE NAVIGATION

INTERNAL PROMOTOR: GEVAERT WOUTER

EXTERNAL PROMOTOR: SAM DEBEUF

RESEARCH CONDUCTED BY

LUCAS DRIESSENS

FOR OBTAINING A BACHELOR'S DEGREE IN

MULTIMEDIA & CREATIVE TECHNOLOGIES

HOWEST | 2023-2024

Abstract

In my research, I explore the exciting world of artificial intelligence, specifically focusing on reinforcement learning (RL) and its practical applications. The central question driving my investigation is whether we can transfer a trained RL agent from a simulation to the real world. This inquiry takes center stage as we delve into the intricacies of maze navigation.

The research is structured around several sub-questions, all aimed at creating a comprehensive understanding of the process. First, I investigate various virtual environments suitable for training a virtual RC car, aiming to find the most effective platform. Next, I identify the best reinforcement learning techniques for this specific application, considering factors like efficiency, adaptability, and real-world applicability. Finally, I explore the challenges involved in bridging the gap between simulation and reality.

Through this study, I hope to contribute significantly to the field of AI and robotics. By offering insights and methodologies, we can potentially advance the implementation of RL in real-world scenarios. The outcomes of this research could have far-reaching implications, not only in robotics but also in other areas where simulation-based training is crucial.

Preface

This bachelor thesis, titled “Exploring the Feasibility of Sim2Real Transfer in Reinforcement Learning,” marks the culmination of my academic journey in New Media & Communication Technology at Howest, University of Applied Sciences. The pivotal question at its core—“Is it possible to transfer a trained RL agent from a simulation to the real world?”—reflects my deep-seated curiosity about the intersection of virtual simulations and tangible applications. It also underscores a broader challenge in AI: creating adaptive, real-world systems from controlled simulations.

My fascination with the concept of Sim2Real transfer originated during the ‘Researchproject’ module. Witnessing the potential of simulated environments to approximate complex real-world behaviors sparked my interest in exploring their practical applicability. In structuring this thesis, I not only probe the theoretical foundations of Sim2Real transfer but also experimentally test its viability through a series of innovative applications. These experiments aim to refine the methods of Sim2Real transfer, enhancing their effectiveness and reliability.

The research methodology adopted for this thesis combines qualitative analyses with quantitative experiments. Theoretical studies provide a comprehensive background, setting the stage for empirical tests conducted in controlled environments. By systematically assessing the performance of RL agents in various scenarios, the research seeks to identify optimal strategies for effective Sim2Real transitions.

I extend heartfelt thanks to my coach and supervisor, Gevaert Wouter, for his invaluable guidance and insights throughout this research. His expertise and mentorship have been fundamental to my scholarly and personal growth. Gratitude is also due to Amaury Van Naemen for his technical support with 3D printing components crucial for my experiments. His assistance was pivotal in navigating the practical challenges of my research.

Additional appreciation goes to the faculty and staff at Howest, whose commitment to fostering an innovative educational environment has profoundly influenced my development. Their unwavering support has been instrumental in my pursuit of technology and innovation.

Lucas Driessens
01-06-2024

Table of Contents

List of Figures

Glossary of Terms

1. **Artificial Intelligence (AI)**: The simulation of human intelligence processes by machines, especially computer systems, enabling them to perform tasks that typically require human intelligence.
2. **Double Deep Q-Network (DDQN)**: An enhancement of the Deep Q-Network (DQN) algorithm that addresses the overestimation of action values, thus improving learning stability and performance.
3. **Epsilon Decay**: A technique in reinforcement learning that gradually decreases the rate of exploration over time, allowing the agent to transition from exploring the environment to exploiting known actions for better outcomes.
4. **Mean Squared Error (MSE)**: A loss function used in regression models to measure the average squared difference between the estimated values and the actual value, useful for training models by minimizing error.
5. **Motion Processing Unit (MPU6050)**: A sensor device combining a MEMS (Micro-Electro-Mechanical Systems) gyroscope and a MEMS accelerometer, providing comprehensive motion processing capabilities.
6. **Policy Network**: In reinforcement learning, a neural network model that directly maps observed environment states to actions, guiding the agent's decisions based on the current policy.
7. **Raspberry Pi (RPI)**: A small, affordable computer used for various programming projects, including robotics and educational applications.
8. **RC Car**: A remote-controlled car used as a practical application platform in reinforcement learning experiments, demonstrating how algorithms can control real-world vehicles.
9. **Reinforcement Learning (RL)**: A subset of machine learning where an agent learns to make decisions by taking actions within an environment to achieve specified goals, guided by a system of rewards and penalties.
10. **Sim2Real Transfer**: The practice of applying models and strategies developed within a simulated environment to real-world situations, crucial for bridging the gap between theoretical research and practical application.
11. **Target Network**: Utilized in the DDQN framework, a neural network that helps stabilize training by providing consistent targets for the duration of the update interval.
12. **Virtual Environment**: A simulated setting designed for training reinforcement learning agents, offering a controlled, risk-free platform for experimentation and learning.
13. **Wheel Slippage**: A phenomenon where the wheels of a vehicle lose traction, causing them to spin without propelling the vehicle forward, often encountered in real-world scenarios with uneven terrain.
14. **Ultrasonic Distance Sensor (HC-SR04)**: A sensor that uses ultrasonic waves to measure distance, commonly employed in robotics for obstacle detection and navigation.

15. **Over the air updates (OTA)**: A method of remotely updating software or firmware on devices, allowing for seamless upgrades and maintenance without physical access to the device.
16. **autonomous vehicle (AV):** A self-driving vehicle capable of navigating and operating without human intervention, relying on sensors, algorithms, and AI to perceive and interact with the environment.

List of Abbreviations

1. **AI** - Artificial Intelligence
2. **DDQN** - Double Deep Q-Network
3. **DQN** - Deep Q-Network
4. **ESP32** - Espressif Systems 32-bit Microcontroller
5. **HC-SR04** - Ultrasonic Distance Sensor
6. **MSE** - Mean Squared Error
7. **MPU6050** - Motion Processing Unit (Gyroscope + Accelerometer)
8. **PPO** - Proximal Policy Optimization
9. **RC** - Remote Controlled
10. **RPI** - Raspberry Pi
11. **RL** - Reinforcement Learning
12. **RCMazeEnv** - RC Maze Environment (Custom Virtual Environment for RL Training)
13. **Sim2Real** - Simulation to Reality Transfer
14. **OTA** Over the air updates
15. **AV** - autonomous vehicle

Chapter 1. Introduction

1.1. Navigating the Maze: Sim-to-Real Transfer in Reinforcement Learning

In our ever-evolving world, the boundaries between virtual simulations and tangible reality are becoming increasingly connected. Imagine a scenario: you, meticulously training a robot within the confines of a computer simulation, now face the daunting task of navigating a physical maze to rescue a stranded hiker. This seemingly straightforward challenge, however, unravels profound questions about the transferability of knowledge from the digital realm to the tangible environment. Welcome to the captivating intersection of **Reinforcement Learning (RL)** and the elusive concept of **sim-to-real transfer.**

1.2. Sim-to-Real Transfer: Bridging the Gap

Sim-to-real transfer - a term that resonates with roboticists, AI enthusiasts, and anyone intrigued by the future—refers to the process of translating learned behaviors from simulated environments into effective actions in the real world. Why does this matter? Because while simulations provide a safe and controlled space for training, they often diverge significantly from reality. Factors like sensor noise, friction, lighting conditions, and unexpected obstacles can confound even the most sophisticated algorithms.

1.3. The Maze Navigation Challenge: RC Cars and Algorithms

The spotlight shines squarely on maze navigation. Imagine an RC car—a miniature explorer—equipped with sensors, wheels, and your curious algorithmic mind. Within the simulated maze, it learns optimal paths, avoids dead ends, and optimizes its trajectory. But can this digital prowess seamlessly translate to the physical maze, where friction, uneven terrain, and unforeseen obstacles await?

1.4. The Expedition: Four Key Steps

1. **Simulator Design:** I embark on creating a realistic maze simulator—one that captures physical nuances like wheel slippage, sensor noise, and limited field of view. The virtual car will explore a maze while learning through trial and error.
2. **Transfer Learning Strategies:** How do I bridge the gap? I'll delve into techniques such as domain adaptation, fine-tuning, and meta-learning. Can we distill the essence of maze-solving without overfitting to the simulation?
3. **Sensor Calibration:** The RC car's sensors—lidar, cameras, and encoders—differ from their virtual counterparts. Calibrating them effectively is crucial. I'll explore sensor fusion and adaptation methods to ensure seamless transitions.
4. **Robust Policies:** The car won't encounter neatly defined corridors; it'll face real-world messiness. Robust policies—resilient to noisy data and unexpected scenarios—are essential.

1.5. Beyond Mazes: A Broader Canvas

While our primary focus remains on mazes, the implications extend far beyond. Imagine autonomous drones navigating urban landscapes, self-driving cars avoiding pedestrians, or medical robots operating in cluttered hospital rooms. Sim-to-real transfer is the bridge that makes these scenarios feasible.

So buckle up (or tighten your wheel nuts), as we embark on this thrilling expedition. The RC car awaits, ready to unravel the mysteries of both simulation and reality.

Chapter 2. Background on Reinforcement Learning

Reinforcement Learning (RL) employs a computational approach where agents learn to optimize their action sequences through trials and errors, engaging with their environment to maximize rewards over time. This learning framework is built upon the foundation of Markov Decision Processes (MDP), which includes:

- **States (S)**: A definitive set of environmental conditions.
- **Actions (A)**: A comprehensive set of possible actions for the agent.
- **Transition Probabilities ($P(s_{t+1}|s_t, a_t)$)**: The likelihood of moving from state s_t to state s_{t+1} after the agent takes action a_t at time t.
- **Rewards ($R(s_t, a_t)$)**: The reward received when transitioning from state s_t to state s_{t+1} due to action a_t.

The principles of Reinforcement Learning, particularly the dynamics of Markov Decision Processes involving states S, actions A, transition probabilities $P(s_{t+1}|s_t, a_t)$, and rewards $R(s_t, a_t)$, form the foundation of how agents learn from and interact with their environment to optimize decision-making over time. This understanding is crucial in the development of autonomous vehicles, improving navigational strategies, decision-making capabilities, and adaptation to real-time environmental changes. The seminal work by R.S. Sutton and A.G. Barto significantly elucidates these principles and complexities of RL algorithms [18].

Chapter 3. Research Questions

This investigation centers around the question: “Can a trained RL agent effectively transition from a simulation to a real-world environment for maze navigation?” To address this question, we’ll explore various aspects of RL training and implementation:

1. **Selecting Virtual Environments**: determine which virtual environments are most effective for RL training.
2. **Suitable RL Techniques**: Identifying RL techniques suitable for autonomous navigation.
3. **Sim-to-Real Transfer Evaluation**: Assessing how well the agent adapts to real-world dynamics.
4. **Simulation-Based Training Efficacy**: Evaluating training effectiveness and optimizing performance through simulation.
5. **Model Adaptation to Real RC Car**: Discussing necessary adjustments for real-world application.

My research combines qualitative and quantitative methodologies, including simulation experiments and real-world trials. By doing so, I aim not only to validate sim-to-real transfer but also to contribute to the ongoing discourse on practical challenges in Reinforcement Learning (RL).

Chapter 4. Methodology

This section explores the Reinforcement Learning Maze Navigation (RCMazeEnv) method, which utilizes a Double Deep Q-Network (DDQNAgent) architecture. We'll delve into the maze environment setup, the design of the DDQN agent, and the comprehensive training algorithm, incorporating mathematical functions to describe the system's mechanics.

4.1. Environment Setup (RCMazeEnv)

The RCMazeEnv is a custom maze navigation environment build upon the OpenAI Gym framework. It's designed for a 12x12 cell grid maze navigation task. Within this grid:

- Cells are either walls (represented by '1') or paths (represented by '0').
- The goal is located at cell position (10, 10).
- The agent, visualized as a car, starts at cell (1, 1) facing eastward.
- The agent can take three possible actions: moving forward, turning left, and turning right.

To aid with navigation, the agent has sensors providing readings in three directions: front, left, and right. These sensors measure the distance to the nearest wall in their respective directions, crucial for decision-making. The environment's state space ($\mathcal{S}$) includes the agent's current position (x, y), orientation θ (north, east, south, or west), and sensor readings $\{s_{\text{front}}, s_{\text{left}}, s_{\text{right}}\}$. The agent's goal is efficient maze navigation, reaching the goal while avoiding collisions with walls or getting stuck in corners all while optimizing its path based on sensor inputs and past experiences.

4.2. Agent Design (DDQNAgent)

The agent uses a Double Deep Q-Network (DDQN) architecture to learn the optimal policy π^*. DDQN is an enhancement over the standard DQN, aiming to reduce overestimation of Q-values by separating action selection from evaluation [19].

- **Policy Network**: Estimates the Q-value $Q(s, a; \theta)$ for taking action a in state s, with weights θ. This network selects actions based on the current policy.
- **Target Network**: Independently parameterized by weights θ^-, it estimates the target Q-value for updating the policy network. The target network mirrors the policy network's architecture but updates less frequently to provide stable target values.

The DDQN update equation modifies the Q-function:

$$Y_t^{DDQN} = R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \theta); \theta^-\right)$$

Where:

- R_{t+1} is the reward received after taking action a in state s.
- γ is the discount factor.
- $\underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \theta)$ selects the action using the policy network.
- $Q\left(S_{t+1}, a; \theta^{-}\right)$ evaluates the action using the target network.

This approach reduces overestimation by separating the max operation in the target, mitigating overoptimism observed in Q-learning [20].

The action space $\mathcal{A}$ and other agent setup details remain consistent. DDQN significantly improves stability and performance by addressing Q-value overestimation, although its effectiveness varies depending on the task compared to traditional DQN approaches [21].

4.3. Training Process

The training process involves utilizing experience replay, where transitions (s, a, r, s') are stored in a replay buffer denoted as D. Our objective is to train a Deep Q-Network (DQN) by minimizing the loss function $L(\theta)$. This loss function quantifies the discrepancy between the current Q-values and the target Q-values:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta)\right)^2\right]$$

Where:

- s represents the current state.
- a corresponds to the action taken.
- r denotes the received reward.
- s' signifies the subsequent state.
- θ^{-} refers to the weights of a target network.
- γ represents the discount factor.

To enhance training stability, I periodically synchronize the target network's weights with those of the policy network. Additionally, we employ an epsilon-greedy strategy for action selection. Initially, we prioritize exploration (with ϵ set to 1), gradually reducing exploration as training progresses. This balance between exploration and exploitation contributes to the DQN's overall performance.

Chapter 5. Reward Function and completion components

In the context of maze navigation, designing an effective reward function is crucial for guiding an agent's learning process. Below, I outline the key components of the reward function used in our framework:

1. **Goal Achievement Bonus (R_{goal}):**

- Reaching the goal is the primary objective of the maze navigation task.
- Upon achieving this objective, the agent receives a substantial reward: $R_{\text{goal}} = +500$.
- However, if the agent takes an excessively long route to reach the goal (more than 1000 steps), it gets a penalty: $R_{\text{goal}} = -200$.
- This mechanism encourages efficient navigation while still rewarding successfully reaching the goal

2. **Proximity Reward ($R_{\text{proximity}}$):**

   - Encourages the agent to minimize its distance to the goal over time.
   - The reward decreases as the distance to the goal increases: $R_{\text{proximity}} = \frac{50}{d_{\text{goal}}+1}$.
   - Here, d_{goal} represents the Euclidean distance to the goal.

3. **Progress Reward (R_{progress}):**

   - Provides immediate feedback based on the agent's movement relative to the goal.
   - If the distance to the goal decreases, the agent receives a positive reward: $R_{\text{progress}} = +50$.
   - if the distance increases, it gets a penalty: $R_{\text{progress}} = -25$.
   - This encourages smarter navigation decisions.

4. **Exploration Penalty (R_{revisit}):**

   - Discourages repetitive exploration of the same areas.
   - The agent receives a penalty for re-entering previously visited cells: $R_{\text{revisit}} = -10$.
   - This promotes exploration of new paths and prevents the agent from getting stuck.

5. **Efficiency Penalty ($R_{\text{efficiency}}$):**

   - Every step the agent takes incurs a small penalty: $R_{\text{efficiency}} = -5$.
   - Balances the need for exploration with the goal of reaching the destination efficiently.

5.1. Termination conditions

To determine whether the environment has reached a "done" or "ended" state, several conditions are established. These conditions include: surpassing 3000 steps, the car being out of bounds (hitting a wall), and the RC car reaching the goal position of $(10, 10)$.

The termination condition can be expressed as:

$$\text{terminate}(steps, position) = \begin{cases} \text{true, "Exceeded max steps"} & \text{if } steps > 3000 \\ \text{true, "Goal reached"} & \text{if } position = (10, 10) \\ \text{true, "Out of bounds"} & \text{if } \neg\text{inBounds}(position) \\ \text{false, "Continue"} & \text{otherwise} \end{cases}$$

5.2. Expanding Real-World Testing

In this study, I conducted experiments indoors to closely replicate theoretical conditions. The tests were performed on a hard cloth surface to minimize ground-related issues and ensure a consistent testing environment. This step was crucial because during real-world testing, the RC car encountered challenges on uneven surfaces.

However, the exploration wasn't limited to indoor setups alone. I also aimed to assess the adaptability and resilience of my proposed solutions in outdoor environments. Taking the experiments outdoors posed significant challenges due to the differences in ground conditions. Outdoor landscapes are diverse and unpredictable, which exposed limitations in my current method's ability to handle such variations. This highlighted the need for further research and improvements in the methods used, such as the hardware limitations.

Chapter 6. Experimental Outcomes and Implementation Details

6.1. Simulation Design and Agent Framework

- **RCMazeEnv**: This environment was specifically designed for the study. It simulates a robotic car navigating through a maze, aiming to replicate real-world physics and limitations. The maze layout, the robotic car's movement capabilities, and its sensor configurations all contribute to creating an authentic testbed for reinforcement learning algorithms.
- **Double Deep Q-Network (DDQN)**: Instead of relying on a single neural network, DDQN employs two networks to enhance standard reinforcement learning techniques. By doing so, it mitigates the overvaluation of Q-values. The policy network and the target network collaborate, continuously interpreting sensor data and improving the learning process.

6.2. Implementation Insights

- **Interaction Between Environment and Agent**: The DDQN agent adapts dynamically to the environment. It relies on sensor feedback to make decisions and fine-tune its path through the maze. This ongoing learning cycle is demonstrated on a simulation platform, allowing us to observe how the agent's tactics evolve over time.
- **Application in the Real World**: Transitioning from virtual training to a physical RC robot involved significant hardware adjustments and fine-tuning. Challenges like aligning sensor data and achieving precise movement control were addressed to ensure a seamless transfer from the virtual model to practical use.

6.3. Performance Evaluation

To evaluate the agent's effectiveness in navigating the maze, I used several metrics such as manually looking at the decision and seeing if the agent would get the car stuck, Reward, loss and epsilon history per episode during training. These metrics provided insights into the agent's learning progress and performance and allowed for quick adjustments to reward functions or hyperparameters.

6.4. Distinctive Elements

- **Physical Maze and Digital Interface**: A real maze was built to mirror the virtual `RCMazeEnv`, which is one of the requirements of ensuring the RC robot's ability to navigate the real maze. In addition, a web application was made to not only control the RC car (starting and stopping movement) but also as a virtual twin. Allowing us to see the sensor readings and decision making in real time.

Chapter 7. Analysis and Results: Addressing Research Questions

7.1. 1. Virtual Environments for RF-Car Training

Selecting an appropriate virtual environment is a crucial initial step in RF-car training. Several platforms, including Unity 3D, AirSim, CARLA, OpenAI Gym, and ISAAC Gym, offer diverse features for driving simulation. However, for RF-car training, I've chosen OpenAI Gym due to its flexibility in creating custom environments and Python compatibility. This choice facilitates seamless integration with existing advanced AI coursework and supports effective SIM2REAL transfer practices [1].

While Unity 3D and AirSim provide realistic simulations, their complexity extends beyond Python, limiting accessibility for my project. CARLA, although comprehensive for autonomous driving simulations, caters more to traditional vehicle models than RF-cars. ISAAC Gym, focused on robotics, also doesn't align perfectly with my goals. OpenAI Gym's simplicity and reinforcement learning focus make it an ideal fit for my project.

7.2. 2. Reinforcement Learning Techniques for Virtual RF-Car Training

Comparing Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and Proximal Policy Optimization (PPO) techniques, I find that DDQN best suits my needs. DDQN's architecture addresses the overestimation bias inherent in DQN, improving Q-value approximation accuracy—an essential factor in navigating complex, sensor-driven RF-car environments.

While DQN excels in high-dimensional sensory input processing, it falls short in unpredictable dynamic environments. DDQN overcomes this limitation. PPO focuses on direct policy optimization but lacks

the precision in value estimation required for RF-car training. Empirical trials confirm DDQN's superior performance in intricate maze-like virtual RF-car scenarios [3].

7.3. 3. Sim-to-Real Transfer Challenges and Solutions

Transferring simulation models to real-world applications involves addressing sensor data interpretation discrepancies, action synchronization, and physical dynamics. I implemented solutions like sensor data normalization and action synchronization mechanisms to align simulation outcomes with real-world performance [5].

Introducing failsafe mechanisms and adjusting motor control timings was something that I overlooked at first, but then it proved to be essential in reducing collision risks and movement inaccuracies during sim-to-real transfer. Iterative testing and adaptation play a vital role in this process [6].

7.4. 4. Contributions of Simulation in RF-Car Training

Simulation training offers efficiency, safety, and computational advantages. It allows uninterrupted, automated training sessions, eliminating real-world risks. Leveraging powerful computing resources accelerates the training process, making simulation indispensable in RF-car development [7].

Comparing simulation and real-world training outcomes highlights the practicality and effectiveness of simulation in developing autonomous driving models.

7.5. 5. Practical Application of Simulated Training to Real-World RF-Cars

Applying a trained model to a physical RC car requires some needed adjustments. Effective sim-to-real adaptation involves fine-tuning sensor interpretations, implementing action synchronization measures, and adjusting physical dynamics to mirror the simulation. This ensures successful application in real-world scenarios, facilitating robust and reliable autonomous driving systems [10].

Chapter 8. Model Architecture and Training Insights

To understand how our Double DQN model learns and makes decisions, let's dive into its architecture. The model consists of four dense layers, which output three actions tailored to the RC car's movement, enabling it to navigate the maze with ease.

Research has shown that, all things being equal, simpler models are often preferred in reinforcement learning [27]. This is because they can lead to better performance, faster learning, and improved generalization. However, finding the right balance of model complexity is crucial [27] [29].. Simplicity is not just about the number of layers or parameters, but also about capturing temporal regularities, such as repetitions, in sequential strategies [27]. [28]..

With these insights in mind, we designed the Double DQN model to strike a balance between simplicity and effectiveness, ensuring optimal performance in maze navigation tasks.

Model Architecture:

```
## Chapter 9. Model: "sequential_52"

# Layer (type) Output Shape Param

=================================================================
dense_200 (Dense) (None, 32) 224
dense_201 (Dense) (None, 64) 2112
dense_202 (Dense) (None, 32) 2080
dense_203 (Dense) (None, 3) 99
=================================================================
Total params: 4515 (17.64 KB)
Trainable params: 4515 (17.64 KB)
Non-trainable params: 0 (0.00 Byte)

---
```

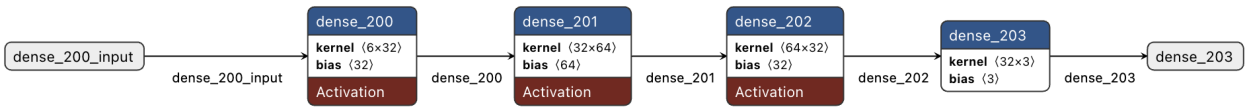


Figure 1: Model Architecture

9.1. Training Parameters

The training of the Double DQN agent was governed by the following parameters:

- **Discount Factor (`DISCOUNT`)**: 0.90
- **Batch Size**: 128
    - Number of steps (samples) used for training at a time.
- **Update Target Interval (`UPDATE_TARGET_INTERVAL`)**: 2
    - Frequency of updating the target network.
- **Epsilon (`EPSILON`)**: 0.99
    - Initial exploration rate.
- **Minimum Epsilon (`MIN_EPSILON`)**: 0.01
    - Minimum value for exploration rate.

- **Epsilon Decay Rate (`DECAY`)**: 0.99973
  - Rate at which exploration probability decreases.
- **Number of Episodes (`EPISODE_AMOUNT`)**: 170
  - Total episodes for training the agent.
- **Replay Memory Capacity (`REPLAY_MEMORY_CAPACITY`)**: 2,000,000
  - Maximum size of the replay buffer.
- **Learning Rate**: 0.001
  - The rate at which the model learns from new observations.

9.2. Training Procedure

1. **Initialization**: Start with a high exploration rate (`EPSILON`) allowing the agent to explore the environment extensively.
2. **Episodic Training**: For each episode, the agent interacts with the environment, collecting state, action, reward, and next state data.
3. **Replay Buffer**: Store these experiences in a replay memory, which helps in breaking the correlation between sequential experiences.
4. **Batch Learning**: Randomly sample a batch of experiences from the replay buffer to train the network.
5. **Target Network Update**: Every `UPDATE_TARGET_INTERVAL` episodes, update the weights of the target network with those of the policy network.
6. **Epsilon Decay**: Gradually decrease the exploration rate (`EPSILON`) following the decay rate (DECAY), shifting the strategy from exploration to exploitation.
7. **Performance Monitoring**: Continuously monitor the agent's performance in terms of rewards and success rate in navigating the maze.

Chapter 10. Visual Insights and Further Exploration

This project takes an innovative approach to sim-to-real transfer in reinforcement learning, which I've showcased through a series of visual elements and demonstrations. These visuals range from the meticulous setup of our physical maze to the user-friendly design of our web application.

- **Maze Visualization:**

The following image provides a more detailed look at the real-world maze setup. This physical representation mirrors the virtual maze environment.

Figure 2: Real life Maze Build

- **Web Application Interface:**

This web application serves as a control interface for the RC car, allowing me to easily monitor what the RC car sees (due to the sensor values being displayed) and emergency stop the car if needed.

![Web App Interface](./images/thesis/web_app.png “Web App (Image created by author)){ width=100% }

- **Simulation Test Video:**

Watch the Double Deep Q-Network (DDQN) in action in this test video. It gives a real sense of how the algorithm navigates through the maze.

- **DDQN Simulation test:** See DDQN Simulation in the Video reference section.

10.1. Evaluation Metrics Overview

10.1.1. Simulation Metrics

Episodic Performance

- **Objective and Goal:** The aim of this metric is to monitor the agent's progress in mastering the maze. By evaluating the learning curve, I see how efficiently the agent can navigate to the end of the maze over successive trials. This gives us insights into its ability to optimize strategies and adapt over time.
- **How it's Assessed:** I measure the number of episodes the agent needs before it can consistently complete the maze. A reduction in episodes over time is a good indicator that the agent is learning and adapting well.
- **Analytical Techniques:** To examine episodic performance, we either conduct statistical analyses or create visual plots, such as learning curves. These tools help us track and visualize changes in performance throughout the training period.
- **Accuracy and Consistency Measures:** To maintain accuracy and consistency, I ensure data integrity and control experimental conditions. Averaging results across multiple trials helps smooth out any randomness in the learning process, providing a clearer picture of the agent's performance.

Step Efficiency

- **Objective and Goal:** This metric evaluates the agent's decision-making efficiency and ability to optimize its path through the maze. By measuring the steps the agent takes to solve the maze, fewer steps indicate a more efficient and smarter learning process.
- **How it's Assessed:** I keep track of the steps required to reach the maze's endpoint in each episode and analyze the reduction in steps over time.
- **Analytical Techniques:** I use quantitative analysis to examine trends in step count. Smoothing techniques may be applied to provide a clearer view of the overarching trends amidst episode-to-episode variability.
- **Accuracy and Consistency Measures:** To ensure reliable metrics, I replicate tests and average results, maintaining the same maze configuration for all experiments.

MSE Loss Measurement

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2$$

- **Objective and Goal:** This metric quantifies the accuracy of the agent's predictions by measuring the squared discrepancies between predicted values and actual outcomes, providing a clear gauge of learning precision.
- **How it's Assessed:** Using the provided mathematical formula, I average the squared differences across all predictions for an episode or series of episodes.
- **Analytical Techniques:** Calculating MSE is straightforward, but understanding its trend requires examining how it correlates with different stages of the agent's learning, such as initial acquisition of knowledge versus later strategy refinement.
- **Accuracy and Consistency Measures:** Regularly evaluating against a validation set or maintaining a consistent testing framework ensures reliable insights into the agent's predictive accuracy and

learning trajectory.

Reward Trend Analysis

- **Objective and Goal:** This analysis helps determine how effectively the agent's actions lead to positive outcomes, which are indicative of its learning and strategy development.
- **How it's Assessed:** By tracking and analyzing the rewards the agent accumulates over time, looking for trends that show an increase in reward collection.
- **Analytical Techniques:** Employing time series analysis or plotting cumulative rewards can vividly illustrate improvements in the agent's decision-making and learning.
- **Accuracy and Consistency Measures:** Averaging trends over several runs and keeping the reward structures consistent throughout the experiments to ensure comparability.

Epsilon Decay Tracking

- **Objective and Goal:** This metric monitors how well the agent balances exploration of new paths with exploitation of known successful strategies, key for adapting learning methods effectively.
- **How it's Assessed:** By observing the decline in the epsilon parameter over episodes, which indicates the agent's shift from exploring to exploiting.
- **Analytical Techniques:** Plotting epsilon values across episodes helps visualize how the agent's learning strategy evolves over time.
- **Accuracy and Consistency Measures:** Applying the epsilon decay strategy uniformly across all training sessions and maintaining consistent experimental conditions to ensure comparability of results.

10.1.2. Real-World Metrics Transitioning to real-world application involved assessing how well the strategies developed in simulation held up when the agent faced a physical maze with real obstacles and constraints.

- **Maze Navigation**: Observing the RC car as it maneuvered through a real-world maze served as direct proof of how effectively the training translated from simulation to reality. This hands-on test demonstrated the practical utility of the trained agent in navigating complex paths.
- **Sensor Data Analysis**: By examining the real-time sensor data during navigation trials, I gained a deeper insight into how the agent interacts with its physical environment. This analysis was crucial for evaluating the agent's ability to avoid obstacles and optimize its pathfinding strategies efficiently.

Chapter 11. Background on Reinforcement Learning Algorithms

11.1. Background on Double Deep Q-Network (DDQN)

The Double Deep Q-Network (DDQN) is an enhancement of the Deep Q-Network (DQN), a pivotal algorithm in the field of deep reinforcement learning that integrates deep neural networks with Q-learning. DQN itself was a significant advancement as it demonstrated the capability to approximate the Q-value function, which represents the expected reward for taking an action in a given state, using high-capacity neural networks.

11.1.1. Evolution from DQN to DDQN **DQN Challenges**: While DQN substantially improved the stability and performance of Q-learning, it was susceptible to significant overestimations of Q-values due to the noise inherent in the approximation of complex functions by deep neural networks. This overestimation could lead to suboptimal policies and slower convergence during training.

DDQN Solution: Introduced by Hado van Hasselt et al., DDQN addresses the overestimation problem of DQN by decoupling the action selection from the target Q-value generation—a technique termed "double learning." In traditional DQN, a single neural network is used both to select the best action and to evaluate its value. DDQN modifies this by employing two networks:

- The **current network** determines the action with the highest Q-value for the current state.
- A separate **target network**, which is a delayed copy of the current network, is used to estimate the Q-value of taking that action at the next state [22].

11.1.2. The Decoupling Effect This separation ensures that the selection of the best action is less likely to overestimate Q-values, as the estimation is made using a different set of weights, thus reducing bias in the learning process. The target network's parameters are updated less frequently (often after a set number of steps), which further enhances the algorithm's stability.

11.1.3. Impact and Applications DDQN has been shown to achieve better performance and faster convergence in complex environments compared to DQN. It is particularly effective in scenarios where precise action evaluation is crucial, such as in video games and robotic navigation tasks. The improved reliability and accuracy of DDQN make it a valuable model for studying reinforcement learning in controlled environments where stability and efficiency are critical.

11.2. Background on Deep Q-Network (DQN)

The Deep Q-Network (DQN) algorithm represents a significant breakthrough in reinforcement learning by combining traditional Q-learning with deep neural networks. This approach was popularized by re-

searchers at DeepMind with their notable success in training agents that could perform at human levels across various Atari games [23].

Core Mechanism: DQN uses a deep neural network to approximate the Q-value function, which is the expected reward obtainable after taking an action in a given state and following a certain policy thereafter. The neural network inputs the state of the environment and outputs Q-values for each possible action, guiding the agent's decisions.

Innovations Introduced:

- **Experience Replay**: DQN utilizes a technique called experience replay, where experiences collected during training are stored in a replay buffer. This allows the network to learn from past experiences, reducing the correlations between sequential observations and smoothing over changes in the data distribution.
- **Fixed Q-Targets**: To further stabilize training, DQN employs a separate target network, whose weights are fixed for a number of steps and only periodically updated with the weights from the training network [23].

11.2.1. DQN Advantages and Applications

DQN's ability to handle high-dimensional sensory inputs directly with minimal domain knowledge makes it highly versatile and effective in complex environments such as video games, where it can learn directly from pixels.

11.3. Background on Q-agent (Q-learning)

Q-agent, based on the Q-learning algorithm, is one of the most fundamental types of reinforcement learning methods. It is a model-free algorithm that learns to estimate the values of actions at each state without requiring a model of the environment [24].

Simplicity and Versatility: Q-learning works by updating an action-value lookup table called the Q-table, which stores Q-values for each state-action pair. These values are updated using the Bellman equation during each step of training based on the reward received and the maximum predicted reward for the next state.

Challenges: While simple and effective for smaller state spaces, Q-learning's reliance on a Q-table becomes impractical in environments with large or continuous state spaces, where the table size would become infeasibly large.

11.3.1. Q-learning Applications

Q-learning has been foundational in teaching agents in environments with discrete, limited state spaces, such as simple mazes or decision-making scenarios with clear, defined states and actions.

11.4. Background on Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a policy gradient method for reinforcement learning that simplifies and improves upon the Trust Region Policy Optimization (TRPO) approach. PPO has become popular due to its effectiveness and ease of use [25].

Optimization Technique: PPO seeks to take the largest possible improvement step on a policy while avoiding

too large updates that might lead to performance collapse. It achieves this through an objective function that includes a clipped term, penalizing changes to the policy that move it too far from the previous policy.

Advantages: PPO is robust to a variety of hyperparameters and can be used in both continuous and discrete action spaces. It has shown great success in environments ranging from simulated robotics to complex game environments.

11.4.1. PPO Applications

PPO is favored in many modern RL applications due to its balance between efficiency, ease of implementation, and strong empirical performance.

11.5. Background on Actor-Critic (AC)

Actor-Critic methods form a broad class of algorithms in reinforcement learning that combine both policy-based (actor) and value-based (critic) approaches [26].

Dual Components:

- **Actor**: Responsible for selecting actions based on a policy.
- **Critic**: Estimates the value function (or Q-value), which is used to evaluate how good the action taken by the actor is.

Advantages: By separating the action selection and evaluation, actor-critic methods can be more efficient than conventional policy-gradient methods. They reduce the variance of the updates and typically converge faster.

11.5.1. Actor-Critic Applications

Actor-Critic algorithms are versatile and can be applied to both discrete and continuous action spaces. They have been effectively used in applications that require balancing exploration of the environment with the exploitation of known rewards, such as in robotics and complex game environments.

Chapter 12. Comparative Analysis of Reinforcement Learning Algorithms in Maze Navigation

In this analysis, we compare various reinforcement learning algorithms, namely Double Deep Q-Network (DDQN), Deep Q-Network (DQN), Q-agent, Actor-Critic (AC), and Proximal Policy Optimization (PPO). This comparison is based on their performance in navigating a complex maze, focusing on efficiency, learning rate, and adaptability.

12.1. Performance Metrics and Visualization

1. Visit Heatmaps

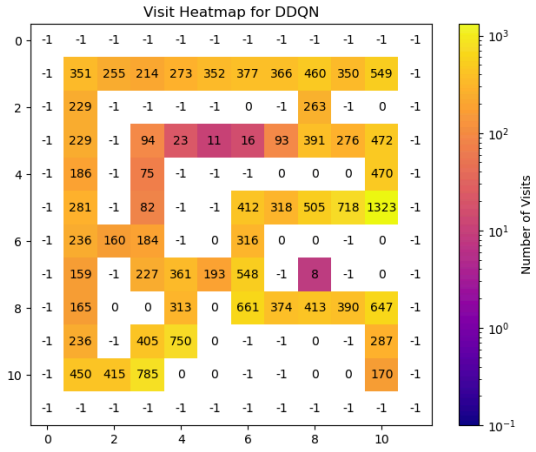


Figure 3: DDQN Heatmap (Image created by author)

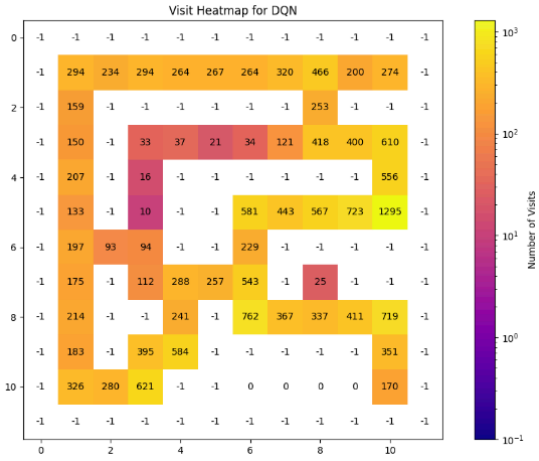


Figure 4: DQN Heatmap (Image created by author)

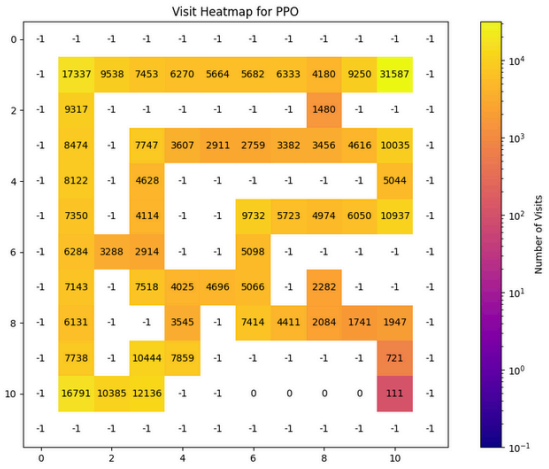
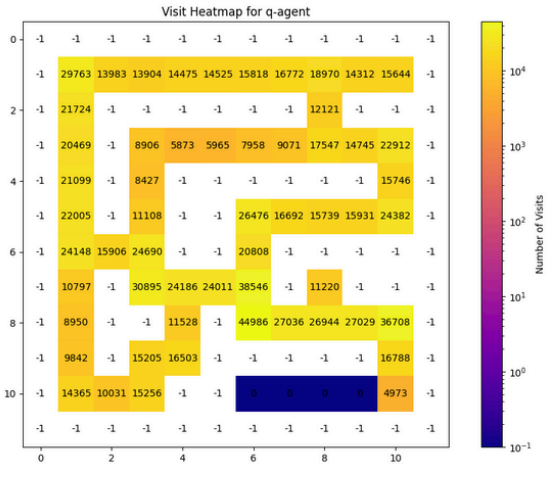


Figure 5: PPO Heatmap (Image created by author)

*



1.2

Figure 6: Q-agent Heatmap (Image created by author)

2. Maze Solution Efficiency

PPO and AC are not included in this visualization due to their relatively higher step counts compared to DDQN and DQN.

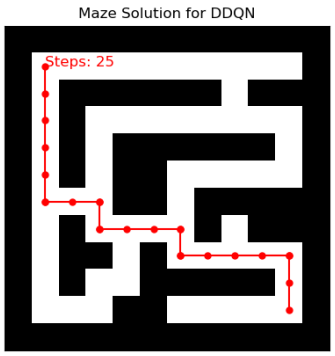


Figure 7: DDQN Maze Path (Image created by author)

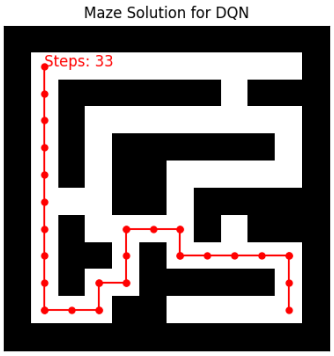


Figure 8: DQN Maze Path (Image created by author)

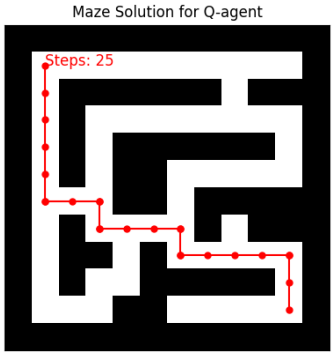


Figure 9: Q-agent Reward History (Image created by author)

3. Reward History and Distribution

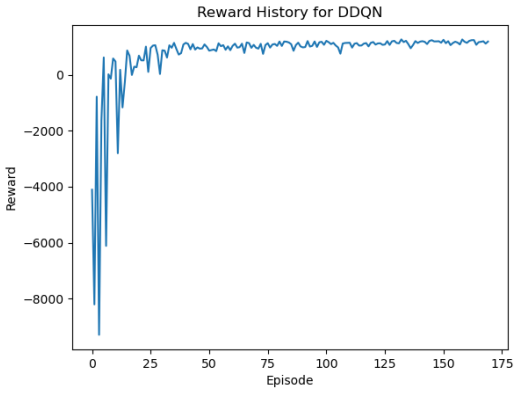


Figure 10: DDQN Reward History (Image created by author)

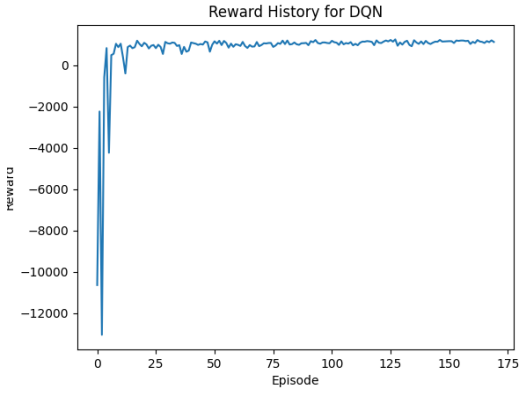


Figure 11: DQN Reward History (Image created by author)

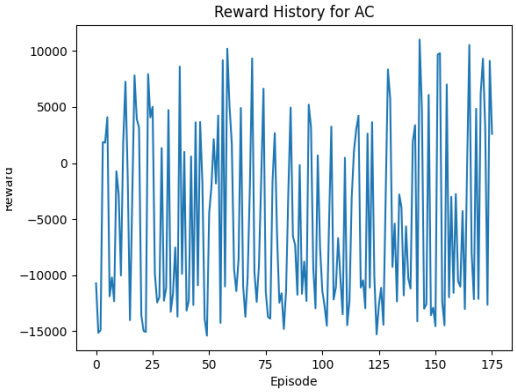


Figure 12: AC Reward History (Image created by author)

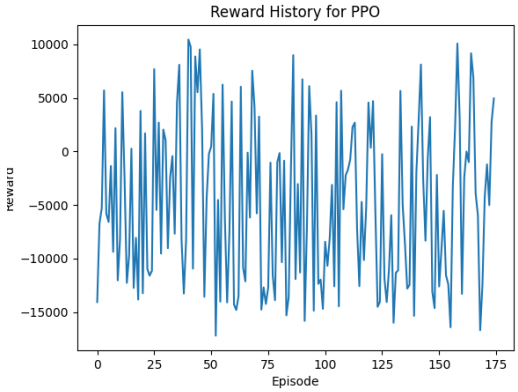


Figure 13: PPO Reward History (Image created by author)

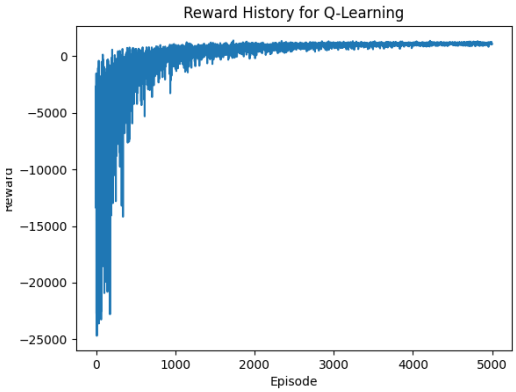


Figure 14: Q-agent Reward History (Image created by author)

4. Mean Squared Error (MSE) Over Time

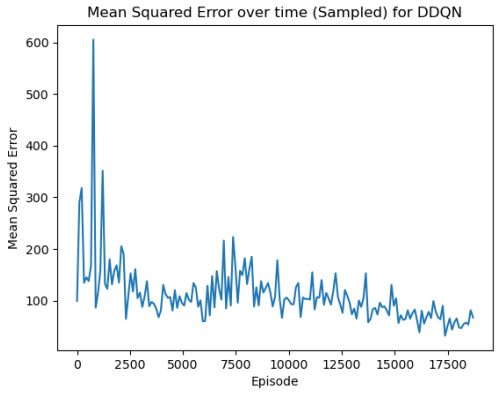


Figure 15: DDQN MSE (Image created by author)

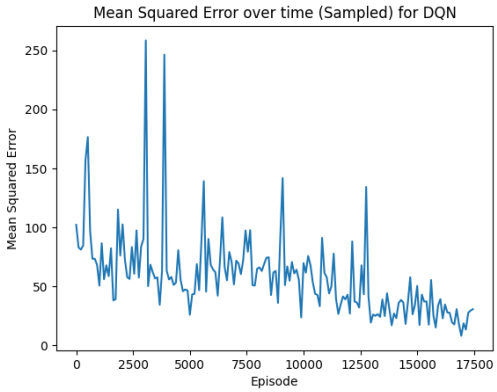


Figure 16: DQN MSE (Image created by author)

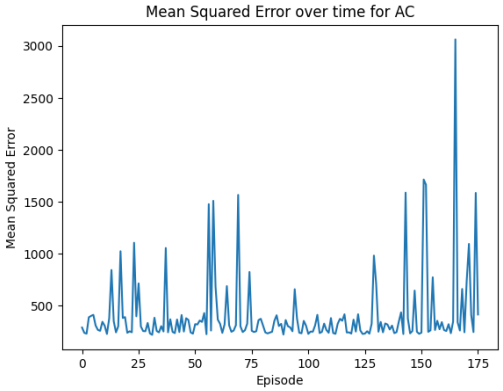


Figure 17: AC MSE (Image created by author)

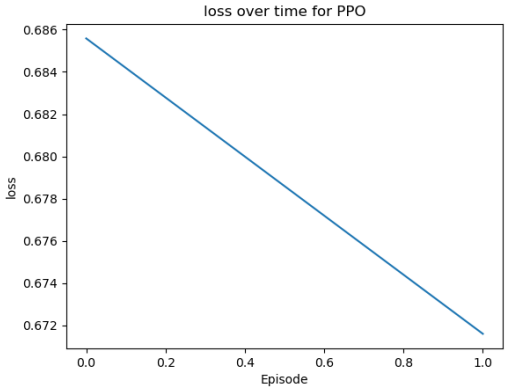


Figure 18: PPO Loss (Image created by author)

5. Moving average of rewards

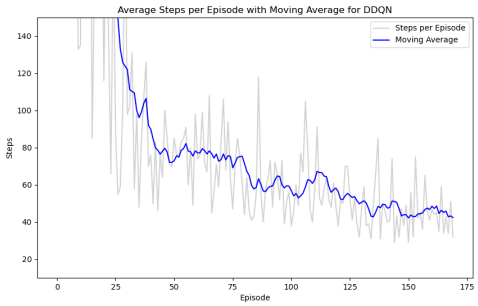


Figure 19: DDQN Moving Average (Image created by author)

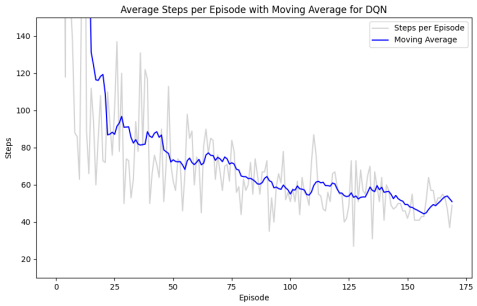


Figure 20: DQN Moving Average (Image created by author)

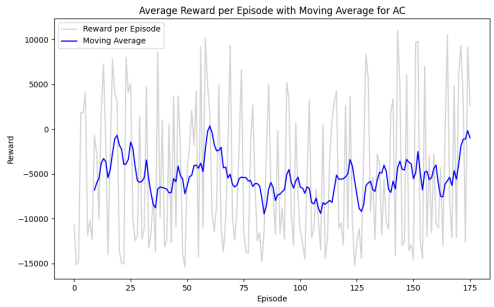


Figure 21: AC Moving Average (Image created by author)

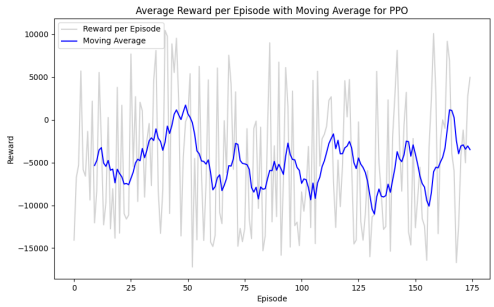


Figure 22: PPO Moving Average (Image created by author)

Chapter 13. Implementation of Real-World Control Algorithms

13.1. Introduction to Real-World Implementation

In this section, I delve into the practical application of control algorithms developed through simulations, now being adapted to control a physical robot. This transition is pivotal for evaluating how simulated behaviors translate into real-world scenarios, thereby assessing the effectiveness and limitations of sim-to-real transfer.

13.2. System Overview

The experimental setup uses an ESP32 microcontroller combined with MPU6050 gyroscopic sensors and ultrasonic sensors for distance measurement, connected to a motor control system. These components enable the robot to perform maneuvers like moving forward, turning left, and turning right. The system's architecture is designed to replicate the simulated environment's dynamics.

13.3. Code Architecture and Integration

System Initialization

Understanding the system's initial setup is crucial for ensuring robust and reliable operation. This phase involves preparing the robot by configuring its hardware interfaces, establishing network connectivity, and setting up sensors and actuators.

- **WiFi and OTA Configuration**: This sets up a network connection and facilitates Over-The-Air (OTA) updates, crucial for remote debugging and iterative improvements.
- **Sensor and Display Setup**: Activates ultrasonic sensors for distance monitoring and initializes a display to provide real-time feedback on the robot's status and IP address, enhancing user interaction and debugging capabilities.

- **MPU6050 Setup and Calibration**: Calibrates the gyroscopic sensor for accurate angle measurements, essential for precise navigation.
- **Motor Setup**: Configures motor drivers and establishes initial motor states, prepping the robot for subsequent movement commands.

Motor Control Mechanism

This subsection elaborates on how movement functions are implemented, translating simulated navigation algorithms into the real-world robotic system.

- **Variables for Motor Control**

  ```
  int initialSpeed = 125; // Higher initial speed for robust movement
  int minSpeed = 40;      // Minimum speed to maintain control
  int speed = initialSpeed;
  constexpr int TURN_DURATION = 245;
  ```

  These variables dictate the motors' initial and minimum speeds, and the duration for turning, facilitating precise and controlled movements by adjusting the speed dynamically based on the robot's turning angle.

- **Forward Movement**

  The `move_forward` function initiates rapid forward motion, with real-time checks for obstacles to ensure safe stops—mimicking the real-world need for dynamic responsiveness.

- **Left Turn**

  The `move_left` function adjusts motor speeds dynamically, a strategy refined in simulations to accommodate physical and inertia effects during turns, ensuring smooth and controlled navigation.

- **Right Turn**

  The `move_right` function applies similar adjustments and sensor feedback to execute precise right turns. Incorporating `calibrateSensors()` before each movement guarantees accurate gyroscopic data, vital for the precise execution of turns.

- **Stopping Movement**

  The `stop_moving` function is designed to immediately halt all motions, crucial for accident prevention and adaptation to sudden changes in dynamic environments.

Calibration and Sensor Data Interpretation

Calibration is crucial for ensuring sensor accuracy and reliability, maintaining the integrity of behaviors developed in simulations when applied in real-world settings. The `calibrateSensors` function periodically recalibrates the gyroscopic sensors to correct any data drift or inaccuracies.

```
void calibrateSensors()
{
    long gyroZAccum = 0;
    Serial.println("Calibrating...");
    for (int i = 0; i < 100; i++)
    {
        int16_t ax, ay, az, gx, gy, gz;
        mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
        gyroZAccum += gz;
        delay(20);
    }
    mpu.setZGyroOffset(-gyroZAccum / 13100); // Calibration based on 100
    ↪ readings
    Serial.println("Calibration Complete");
}
```

13.4. Real-World Application and Limitations

Transitioning from simulated environments to real-world applications presents a complex set of challenges, especially regarding sensor data interpretation and vehicle movement replication. This section explores these critical aspects, emphasizing both opportunities and constraints in applying simulation-derived insights to actual autonomous vehicle (AV) operations.

13.4.1. Enhanced Sensor-Based Navigation Sensor-based navigation technologies, refined through simulation, promise significant improvements in the functionality of autonomous vehicles. In real-world applications, such technologies are crucial for environments demanding high precision and adaptability. For example, in congested urban settings or automated delivery systems, dynamic navigation with high accuracy can enhance both safety and efficiency. By integrating insights from simulations, sensor-based navigation systems can be better tuned to interpret the complex and variable conditions of the real world.

13.4.2. Informing Autonomous Vehicle Movement Simulated environments provide a controlled setting to study vehicle dynamics and movement responses. Applying these insights to the development of autonomous vehicles can lead to advanced algorithms capable of handling the unpredictable nature of real-world environments. This knowledge is instrumental in enhancing autonomous systems' ability to safely and efficiently navigate through dynamic and often chaotic traffic conditions, improving the overall functionality of autonomous transportation.

13.5. Limitations

13.5.1. Discrepancies in Sensor Data Interpretation A significant challenge in the real-world application of simulation-based insights is the variation in sensor data accuracy between simulated and actual environments. These discrepancies can directly impact the effectiveness of navigational algorithms, potentially compromising the vehicle's decision-making processes, safety, and operational efficiency.

13.5.2. Insufficient Data for Exact Positioning One of the most critical limitations encountered was the inability of the sensors to provide enough data to determine the car's exact position within the real-world grid. In the simulation, sensors provided precise, reliable data, enabling accurate positioning and movement control. However, in real-world conditions, the sensor data was often noisy, incomplete, and affected by various environmental factors.

- **Limited Sensor Resolution**: The sensors, such as ultrasonic distance sensors, had limited resolution and range, which affected their ability to detect fine details necessary for precise positioning.
- **Environmental Noise**: Real-world environments introduced noise from various sources such as other electronic devices, surface irregularities, and ambient conditions, which interfered with sensor readings.
- **Dynamic Obstacles**: Unlike the controlled simulation, real-world environments had dynamic obstacles that the sensors could not always accurately detect or interpret, leading to positioning errors.

These limitations meant that the vehicle often had a rough estimate of its position rather than a precise one, which hindered its ability to navigate accurately within a defined grid.

13.5.3. Challenges in Movement Replication Replicating precise vehicle movements from simulated to real-world conditions encounters numerous obstacles. External factors such as road surface variations, environmental conditions, vehicle load, and mechanical constraints can introduce unforeseen deviations in vehicle behavior. These real-world variances necessitate adjustments and recalibration of the algorithms developed in simulated environments to ensure their effectiveness and reliability outside the lab.

13.5.4. Practical Implementation Considerations Successfully translating simulation insights into real-world applications requires meticulous attention to several practical aspects. These include sensor calibration to account for environmental influences, adapting algorithms to hardware limitations, and ensuring the system's resilience to real-world unpredictabilities. Addressing these factors is crucial for the effective deployment and operational success of autonomous vehicles based on simulation-to-reality (sim2real) insights.

13.6. Conclusion for Real-World Application

Transitioning from simulation-based research to practical real-world applications in autonomous vehicle navigation presents unique challenges and opportunities. While applying simulation-derived insights into sensor use and vehicle movement has the potential to revolutionize autonomous vehicle technologies, significant effort is required to bridge the gap between simulated accuracy and real-world variability. Overcoming these challenges is essential for the successful integration of sim2real technologies in enhancing the safety, efficiency, and reliability of autonomous transportation systems.

Chapter 14. Challenges and Solutions in RL Implementation

14.1. Challenge 1: Choosing the Right Virtual Environment

- **Description**: Selecting a suitable virtual environment for training the RC car.
- **Solution**: I chose **OpenAI Gym** due to its simplicity, familiarity from previous coursework, and its focus on reinforcement learning.

14.2. Challenge 2: Selecting the Optimal Reinforcement Learning Technique

- **Description**: Finding the best RL technique for training the virtual RC car.
- **Solution**: Through comparative analysis and testing, the Double Deep Q-Network (DDQN) emerged as the best technique, consistently solving the maze in fewer steps and episodes than other methods.

14.3. Challenge 3: Addressing Movement Discrepancies in Sim2Real Transfer

- **Description**: Bridging the gap between simulation and real-world RC car movement and control.
- **Solution Attempt**: I fine-tuned the frequency of action commands using an async method, waiting for the motor to finish moving, or considering a queued action system. Precise movement in the real world was highlighted as more critical compared to the simulation.

14.4. Challenge 4: Alignment Issues and Motor Encoder Implementation

- **Description**: Achieving precise straight-line movement in the RC car, with a persistent ~3-degree offset.
- **Solution Attempts**:
    - **Attempt 1**: Implemented motor encoders to improve movement accuracy but faced limitations in precision.

- **Attempt 2**: Replaced the motor with a more powerful one, but the added weight led to the same alignment issue.
- **Attempt 3**: Used an MPU6050 gyroscope to measure and adjust the car's orientation, achieving accurate 90-degree turns initially but failing to solve the ~3-degree offset issue.
- **Attempt 4**: Removed the Raspberry Pi and used only the ESP32 to control all sensors and motors, resulting in a lighter robot that moved forward more precisely but still struggled with consistent 90-degree turns.

14.5. Challenge 5: Ensuring Consistent and Effective Training

- **Description**: Maximizing training efficiency and performance while maintaining consistency between simulation and real-world scenarios.
- **Solution**: Training in a simulation proved much more efficient due to the challenges of resetting the RC car, manual interferences, and limited battery life.

14.6. Challenge 6: Accurate Sensor Data Normalization for Sim2Real Transfer

- **Description**: Aligning sensor data between simulated and real-world environments for model accuracy.
- **Solution**: Implemented functions to ensure sensor data fed into the agent matched the data it was trained on.
  - **Real-World Sensor Data Normalization**:

$$\text{map\_distance}(d) = \begin{cases} d & \text{if } d < 25 \\ 25 + (d - 25) \times 0.5 & \text{otherwise} \end{cases}$$

  - **Simulation Sensor Data Normalization**:

$$\text{normalize\_distance}(d) = \max\left(0, \min\left(\frac{d}{\text{sensor\_max\_range}}, 1\right)\right) \times 1000$$

14.7. Challenge 7: Integration of Failsafe Mechanisms

- **Description**: Preventing potential collisions and ensuring safe navigation in the real world.
- **Solution**: Developed a failsafe system to prevent unwanted forward movement, retraining the model with this addition to solve the issue of the robot driving into walls and getting stuck.

14.8. Challenge 8: Training Environment and Technique Efficacy

- **Description**: Determining the most effective environment and RL technique for training.

- **Solution**: The DDQN technique was more efficient than DQN, Q-agent, PPO, and ActorCritic approaches, highlighting the importance of technique selection.

14.9. Conclusion for Challenges and Solutions

This section has outlined the practical challenges encountered in applying reinforcement learning (RL) techniques to autonomous RC cars. My journey began with selecting OpenAI Gym as the virtual environment due to its simplicity and relevance to RL. The Double Deep Q-Network (DDQN) emerged as the most effective RL technique for navigating complex environments.

However, transitioning from simulated models to real-world applications revealed significant discrepancies, particularly in movement control and sensor data alignment. I explored innovative solutions such as motor encoders, power adjustments, and gyroscope integration, which partially addressed these issues. Efforts to normalize sensor data and implement failsafe mechanisms also contributed to better alignment with real-world conditions.

A significant advancement was achieved by simplifying the robot's design to use only the ESP32 module, making it lighter and more precise. This change marked a considerable step in overcoming the challenges previously faced.

Although substantial progress was made, some challenges remain, indicating the need for ongoing research and development to fully harness the potential of RL in autonomous RC car navigation.

Chapter 15. Integration of Practical Experiments

Throughout my research, I used various practical experiments to solve the challenges I encountered. These experiments, documented through video demonstrations, provide clear insights into my problem-solving process.

15.1. Addressing Alignment and Orientation Challenges

One of the main challenges was ensuring the RC-car's precise orientation and alignment during movement. To address this, I used the MPU6050 gyroscope to correct alignment issues and achieve accurate 90-degree turns. My efforts focused on using the gyroscope to maintain and correct the car's orientation, crucial for navigating complex mazes with high precision.

15.1.1. Using the MPU6050 Gyroscope for Precise Orientation

- **Experiment E1 - Gyroscope Calibration**: Testing the MPU6050 gyroscope to correct the car's orientation for accurate navigation, aiming to improve control over the vehicle's movement through maze environments (see Video E1 in the Video References section).

- **Experiment E2 - Navigational Corrections**: Addressing alignment issues for precise 90-degree turns and realigning the car's forward movement to fix a persistent ~3-degree offset (see Video E2 in the Video References section).

15.2. Improving Movement Precision with Encoders

To enhance the RC-car's movement precision, I experimented with rotary encoders. These devices, which accurately measure wheel rotations, were essential for improving straight-line movements and addressing hardware reliability challenges in real-world applications.

- **Experiment E6 - Encoder Implementation**: Adding rotary encoders to the setup to gain more precise control over the car's movements by accurately measuring wheel rotations, thus refining the vehicle's navigation capabilities (see Video E6 in the Video References section).
- **Experiment E7 - Troubleshooting Encoder Malfunction**: Addressing a malfunction with one of the encoders that halted further tests, highlighting the practical challenges of maintaining hardware reliability (see Video E7 in the Video References section).

15.3. Real-World Application Tests

Moving beyond controlled environments, I tested the RC-car in both outdoor and indoor settings to evaluate its performance in real-world conditions. These tests were crucial for assessing the practical application of my research findings and understanding the challenge of accurately translating simulation models to real-world applications.

- **Experiment E9 - Outdoor Navigation Test**: Navigating the RC-car on uneven outdoor surfaces, where variations greatly affected performance, underscoring the importance of environmental factors in autonomous navigation (see Video E9 in the Video References section).
- **Experiment E11 - Indoor Controlled Test**: Conducting controlled indoor tests to closely monitor and adjust the RC-car's navigation strategies, reflecting on the complexities of sim-to-real transfer (see Video E11 in the Video References section).

Chapter 16. Reflection

Reflecting on this research project, I've gained essential insights that have profoundly influenced my approach and perspective.

One of the key lessons learned is the value of openness. Embracing new ideas and venturing beyond familiar territory was crucial for fostering creativity and discovering innovative solutions. Moving forward, I am committed to maintaining this exploratory spirit, keen to harness the winds of novelty.

Bridging the gap between theory and practice proved challenging. While the virtual environments were controlled, the real world demanded adaptability to unforeseen complexities. This experience has sharpened my ability to navigate between the elegance of theory and the unpredictability of real-world applications.

Overcoming barriers taught me the importance of anticipatory thinking. From navigating corporate hurdles to adapting to evolving safety standards, looking ahead has become an integral part of my approach, transforming potential obstacles into opportunities for innovation.

Engaging with policy and regulatory frameworks highlighted the delicate balance between innovation and legislation. Working with policymakers and industry leaders, I learned the importance of crafting regulations that foster innovation while ensuring public safety and accountability.

The societal implications of autonomous systems have been profound. From enhancing mobility for the disabled to influencing urban planning, the potential for positive impact is immense. I am more aware of the need to ensure these technologies are accessible and beneficial across societal divides.

As I move forward, my ethos will be defined by adaptability, responsiveness, and a commitment to societal stewardship. The lessons learned have prepared me for the next stage of my journey, where I will continue to engage with data, refine methodologies, and embrace the challenges that lie ahead.

Insights from interviews provided valuable perspectives on specific technical aspects. For instance, choosing an RC car that can be extensively customized and programmed is crucial for a project focused on sim-to-real transfer. A model with a robust and accessible API, compatibility with various sensors, and the ability to handle different terrains would allow for more detailed control algorithms and a better understanding of how physical properties affect simulation results. Additionally, virtual environments like Unity 3D coupled with ROS offer extensive freedom for simulation, supporting a wide range of movement behaviors and interactions that can be programmed and tested in detail before real-world deployment.

Employing a virtual twin can offer significant advantages over traditional camera systems, especially in terms of the depth of simulation and pre-testing. This approach can lead to better anticipation of how the car would behave in the real world under various conditions, thereby optimizing the algorithms more effectively before actual implementation. However, using a camera for real-time feedback and adjustments based on visual data is invaluable, so ideally, both approaches would be integrated for the best results.

The project can be further developed by integrating more complex sensory systems, applying more sophisticated machine learning models, extending the testing environments to include varying conditions, and fostering community collaboration. These enhancements could not only improve the technology but also advance the field of autonomous vehicle research and development.

Reflecting on the path this research has taken, from concept to implementation, I've gathered key takeaways. Setting clear success criteria at the outset provided direction and motivation. Achieving control over the RC car was gratifying, demonstrating the effectiveness of the reinforcement learning techniques. However, consistency in real-world application remained challenging, highlighting the need for ongoing refinement and adaptation of the algorithms used.

Feedback from seasoned experts was invaluable, helping refine the project by enhancing sensor capabilities and adjusting control algorithms. Ethical considerations were paramount, emphasizing the need for responsible innovation in areas like privacy, safety, and the impact of automation on employment.

This project has implications far beyond the academic realm, influencing policy, industry practices, and public perceptions. It has sparked important discussions that will hopefully lead to more informed and ethical technology development. As this project closes, new questions arise, setting the stage for further exploration of how to balance progress with ethical considerations. I am eager to continue this journey, guided by the insights gained and motivated by both the achievements and the challenges that remain.

Chapter 17. Advice

17.1. Practical Utilization of Simulations

Simulations are invaluable in research, offering a risk-free, controllable environment for developing and refining algorithms.

- **Cost-Effectiveness:** Simulations allow for substantial cost savings by reducing the need for physical prototypes and extensive real-world trials in the early phases of research.

17.2. Strategies for Effective Transition from Simulation to Reality

Successfully transitioning from simulations to real-world applications is critical for validating the effectiveness of research outcomes.

- **Incremental Testing:** Start with simulations to refine algorithms, then gradually introduce real-world testing to confirm results and adapt to environmental variables.
- **Feedback Loops:** Use continuous feedback mechanisms to enhance simulation models with insights gained from real-world tests, improving their accuracy and applicability.

17.3. Overcoming Common Challenges in Simulation-to-Reality Transitions

Bridging the gap between simulations and actual conditions often requires specific adjustments, especially in terms of sensor data and mechanical operations.

- **Sensor Discrepancy Adjustments:** Regular calibration of real-world sensors is essential to ensure they align with simulation inputs.
- **Movement and Mechanics Alignment:** It's crucial to synchronize physical movements and mechanics with those anticipated by simulations to ensure smooth transitions.

17.4. Insights from My Research

- **Simulation Platforms:** Choosing the right simulation platform, like OpenAI Gym, is critical and may necessitate additional tools for more complex scenarios.
- **DDQN Superiority:** My findings show that Double Deep Q-Network (DDQN) surpasses other models, such as DQN and PPO, by reducing overestimations and enhancing learning stability.

17.5. Methodological Advice

- **Comprehensive Evaluation:** Utilize a mix of qualitative and quantitative approaches to thoroughly evaluate the effectiveness of both simulations and real-world applications.

- **Adaptive Techniques:** Stay flexible and responsive to the results and feedback, which are crucial for effectively tackling unexpected challenges.

17.6. Practical Experiment Integration

- **Prototyping and Iteration:** Use iterative design and prototyping to progressively refine systems, effectively linking theoretical research and practical implementation.
- **Continuous Feedback:** Continuously seek and integrate feedback from stakeholders and peers to improve simulation models and real-world applications.

17.7. Guidelines for Future Research

17.7.1. Introduction for Future Research This chapter outlines a detailed methodology and provides advice for researchers engaged in simulation-based studies, aimed at ensuring a successful transition from theoretical models to practical applications.

17.7.2. Step-by-Step Plan

Step 1: Selection of Simulation Environments

- **Research and Evaluation:** Explore and evaluate available simulation tools suited to your study, such as OpenAI Gym, Unity 3D, and CARLA.
- **Criteria Development:** Define criteria focusing on fidelity, scalability, and integration capabilities.
- **Preliminary Testing:** Conduct initial tests to assess the environments against these criteria.

Step 2: Managing Expectations and Adaptability

- **Expectation Setting:** Establish realistic expectations for the capabilities of simulations.
- **Adaptation Strategies:** Be prepared to modify your research approach based on outcomes from simulations and data discrepancies.

Step 3: Methodology Flexibility

- **Continuous Evaluation:** Consistently re-evaluate the effectiveness of your methodologies.
- **Integration of New Technologies:** Embrace emerging technologies as they become relevant to enhance and expand your research.

Chapter 18. Sources of Inspiration and Conceptual Framework

The inspiration for this research draws from a diverse collection of sources, uniquely combining insights from technical documentation, digital platforms, and academic literature. Central to the inspiration were the challenges of micro mouse competitions and the potential of reinforcement learning (RL) in navigating these complex mazes. These initial sparks of interest were further fueled by dynamic demonstrations of RL applications in autonomous vehicle control, particularly through the lens of YouTube and GitHub repositories, alongside influential academic research.

18.1. Micro mouse Competitions and Reinforcement Learning

Micro mouse competitions, which task small robotic mice with the navigation of mazes, served as a foundational inspiration for this study. The direct application of RL in these competitions and related technological showcases provided a compelling narrative on the potential of RL in real-world problem-solving and autonomous control. The exploration of maze traversal algorithms and the strategies for shortest path finding, as detailed in the insightful Medium article by M. A. Dharmasiri [15], enriched the conceptual foundation by illustrating practical algorithmic approaches in similar contexts.

18.2. Influential YouTube Demonstrations and GitHub Insights

YouTube videos such as “Self Driving and Drifting RC Car using Reinforcement Learning” [11] and “Reinforcement Learning with Multi-Fidelity Simulators – RC Car” [16] provided vivid demonstrations of RL’s applicability in real-world settings, emphasizing the feasibility of sim-to-real transfer. These resources, along with GitHub repositories detailing ventures like the “Sim2Real_autonomous_vehicle” project [13], highlighted the practical steps and challenges in implementing RL in physical systems.

18.3. Technical Exploration and Academic Foundation

The academic exploration was significantly shaped by articles on autonomous driving decision control by Q. Song et al.[12] and a survey on sim-to-real transfer in deep reinforcement learning for robotics by W. Zhao, J. P. Queralta, and T. Westerlund [17], which detailed the application of advanced RL algorithms in controlling autonomous vehicles. These articles provided a deep dive into the methodologies and challenges of applying RL in autonomous systems, offering a broad academic perspective on the field.

18.4. Synthesis and Research Direction

These varied sources collectively informed the development of this research, steering the focus towards the feasibility and intricacies of sim2real transfer in the realm of autonomous navigation. The exploration

aims to synthesize insights from both digital and academic realms, tackling the nuanced challenges of applying sophisticated RL models in practical, tangible scenarios.

Chapter 19. General Conclusion

This thesis has effectively demonstrated the potential of transferring a trained reinforcement learning (RL) agent from a simulated environment to a real-world setting, focusing specifically on navigating a maze using a remote-controlled (RC) car. The detailed experiments and analyses discussed in earlier chapters offer a comprehensive exploration of this transition.

The research conclusively shows that such a transfer is not only possible but also fraught with significant challenges. The critical experiments detailed in **Chapter 7: Analysis and Results: Addressing the Research Questions** highlight the importance of normalizing sensor data and adapting control algorithms to handle the unpredictable dynamics of the real world. These adaptations were crucial for aligning the simulated models with the real-world scenarios encountered during implementation.

The selection of appropriate virtual environments and reinforcement learning techniques, as discussed in **Chapter 4: Methodology**, played a key role in shaping the experimental approach and ensuring the effectiveness of the simulation training. The Double Deep Q-Network (DDQN) emerged as the most suitable technique, providing a robust framework to navigate the complexities of practical applications.

This study not only confirms the feasibility of sim-to-real transfers but also provides a detailed examination of the intricate mechanics involved in this process, an area of growing importance in AI and robotics research. By integrating theoretical insights with practical applications, this thesis makes a significant contribution to the ongoing discourse on the viability and challenges of applying reinforcement learning in real-world scenarios.

In conclusion, while it is feasible to transition a trained RL agent from simulation to a real environment, the process requires careful planning, adaptability, and continual refinement. The challenges highlighted throughout this research underscore the need for ongoing efforts to enhance the robustness and reliability of sim-to-real applications, ensuring they can meet the demands of real-world conditions.

Chapter 20. Guest Speakers

20.1. Innovations and Best Practices in AI Projects by Jeroen Boeye at Faktion

Jeroen Boeye's talk, delivered on behalf of Faktion, provided valuable insights into the close relationship between software engineering and artificial intelligence in developing AI solutions. He emphasized the importance of not just focusing on AI technology but also on the software engineering principles that support the creation of robust, scalable, and maintainable AI systems. This approach ensures that AI solutions are both technically sound and viable for long-term application.

During his lecture, Jeroen highlighted several aspects of AI application, notably Chatlayer's impact on conversational AI. He explained how Chatlayer improves chatbot interactions through sophisticated conversational flows, enhancing the accuracy and relevance of exchanges with users. Another point of discussion was Metamaze, which he commended for its innovative methods in automating document processing, creating succinct summaries from extensive documents and emails, showcasing the capabilities of supervised machine learning in administrative tasks.

Jeroen outlined a clear roadmap for successful AI project implementation, stressing the need to validate business cases and adopt a problem-first strategy. He discussed the critical role of high-quality data as the foundation for any AI endeavor and offered strategies for creatively overcoming data limitations. The talk also covered the importance of embracing failures as opportunities for innovation and maintaining open communication with stakeholders about challenges and setbacks.

The lecture further presented various practical AI applications across different industries, such as solar panel detection, unauthorized pool identification, air freight container inspection, and early warning systems for wind turbine gearboxes. Jeroen demonstrated how AI could tackle complex challenges through innovative data sourcing, synthetic data generation, and anomaly detection techniques. He also explored case studies on energy analysis in brick ovens and egg incubation processes, emphasizing the importance of data preprocessing and machine learning models in improving efficiency and outcomes.

Key points from Jeroen's talk included the mastery of data preprocessing and treating data as a dynamic asset to better tailor AI models to specific needs. He shared practical tips on enhancing operational efficiency, such as using host mounts for code integration and Streamlit for dashboard creation, to streamline development processes.

In summary, Jeroen Boeye's lecture offered a thorough perspective on integrating AI technologies in real-world settings. His insights into the vital role of software engineering principles, alongside a deep understanding of AI capabilities and constraints, provided valuable guidance for developing effective and sustainable AI solutions. The lecture not only underscored current AI trends and future directions but also shared practical knowledge on navigating the complexities of AI project execution.

20.2. Pioneering AI Solutions at Noest by Toon Vanhoutte

Toon Vanhoutte's engaging lecture, on behalf of Noest from the Cronos Group, shed light on the effective integration of artificial intelligence and software engineering in developing cutting-edge business solutions. With a dedicated team of 56 local experts, Noest has built a reputation for its pragmatic approach to projects, targeting global impact while valuing craftsmanship, partnership, and enjoyment as core principles. This philosophy extends to their diverse services, which include application development, cloud computing, data analytics, AI innovations, low-code platforms, ERP solutions, and comprehensive system integrations, all supported by a strong partnership with Microsoft.

Toon presented a case study on a packaging company that aimed to revolutionize image search capabilities based on product labels. The project faced various challenges, such as inconsistent PDF formats and large file sizes, which were adeptly managed using Azure Blob Storage for data handling and event-driven processing strategies for efficient, cost-effective solutions, showcasing Noest's skill in utilizing cloud technologies to address complex issues.

Another significant challenge was enhancing image searchability, which involved recognizing text and objects within images. This was tackled using Azure AI Search, supplemented by Large Language Models (LLMs) and vector search techniques. This approach allowed for nuanced search functionalities beyond simple text queries, showing the advanced capabilities of AI in interpreting complex data.

Toon also explored advancements in semantic search, discussing how different search methods—keyword, vector, and hybrid—along with semantic ranking, could significantly improve the accuracy and contextuality of search results. Practical demonstrations, including comparisons between OCR and GPT-4 vision, illustrated the potential of AI to offer deeper insights based on semantic understanding.

A key takeaway from Toon's lecture was the importance of setting realistic client expectations regarding AI's capabilities and potential inaccuracies, highlighting the experimental nature of these technologies. The discussion on AI's evolving landscape emphasized the need for prompt engineering, the challenges of navigating a developing field, and the importance of client education in managing expectations about AI technologies like GPT.

In conclusion, Toon Vanhoutte's presentation not only highlighted Noest's innovative work in AI and software engineering but also imparted crucial lessons on innovation, adaptable problem-solving, and the necessity for ongoing learning in the dynamic field of AI. This presentation showcased Noest's commitment to pushing technological boundaries to create impactful, pragmatic solutions that fully utilize AI's potential.

Chapter 21. Installation Steps

This section outlines the required steps to install and set up the project environment. Adherence to these instructions will ensure the successful deployment of the autonomous navigation system.

21.1. Prerequisites

Before initiating the setup process, ensure the following prerequisites are met:

- **Git:** Necessary for cloning the project repository.
- **Docker:** Utilized for containerizing the web application and ensuring a consistent runtime environment.
- Optionally, **Python 3.11** and **pip** may be installed along with the dependencies listed in `requirements.txt` for running the project without Docker.

21.2. Repository Setup

To clone the repository and navigate to the project directory, execute the following commands:

```
git clone https://github.com/driessenslucas/researchproject.git
cd researchproject
```

21.3. Hardware Setup and Assembly

21.3.1. Introduction to Hardware Components This section provides an overview of the hardware components used in the project, including the RC car, sensors, and microcontrollers. The integration of these components is essential for the successful implementation of the autonomous navigation system.

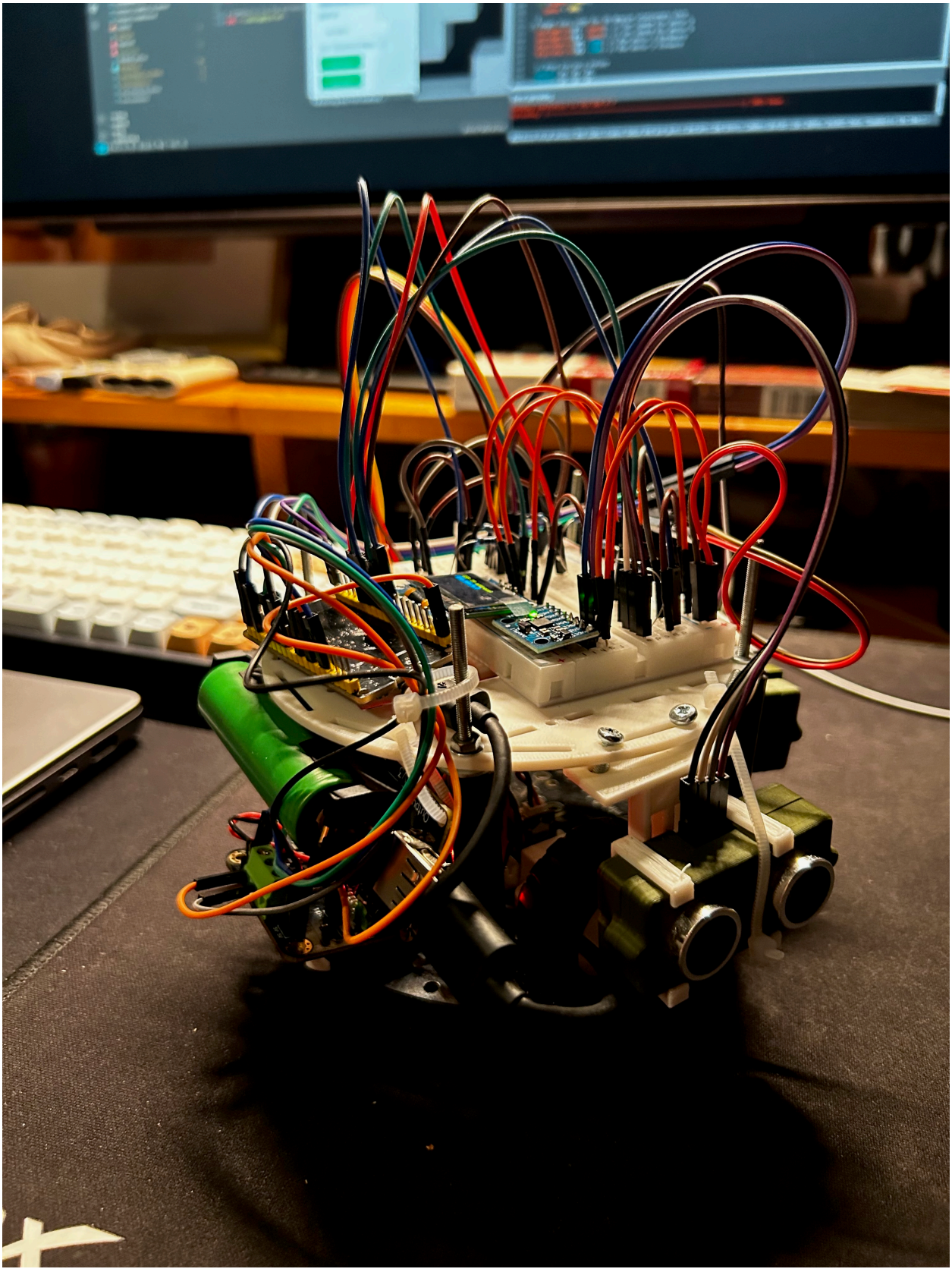


Figure 23: Final RC car

21.3.2. Components List

- **Core Components**:
  - ESP32-WROOM-32 module (Refer to the datasheet at Espressif)
  - 3D printed parts from Thingiverse (hc-sr04, top plate + alternative for the robot kit)
  - Motor Driver - available at DFRobot
  - 2WD robot kit - available at DFRobot
  - Mini OlED screen - available at Amazon
  - Sensors - available at Amazon
  - Battery For ESP 32 - available at Amazon
- **Supplementary Materials**: List of additional materials like screws, wires, and tools required for assembly.
  - 4mm thick screws 5mm long to hold the wood together - available at brico
  - m3 bolt & nuts - available at brico
  - wood for the maze - available at brico

21.3.3. Wiring Guide esp32 pins

```
int E1 = 2; //PWM motor 1
int M1 = 17; //GPIO motor 1
int E2 = 19; //PWM motor 2
int M2 = 4; //GPIO motor 2

int sensor0Trig = 27; //GPIO right sensor
int sensor0Echo = 26; //GPIO right sensor

int sensor1Trig = 33; //GPIO left sensor
int sensor1Echo = 32; //GPIO left sensor

int sensor2Trig = 25; //GPIO front sensor
int sensor2Echo = 35; //GPIO front sensor

// OLED display and MPU6050 pins
#define SDA_PIN 21 // this is the default sda pin on the esp32
#define SCL_PIN 22 // this is the default scl pin on the esp32
```

ESP32 Wiring:

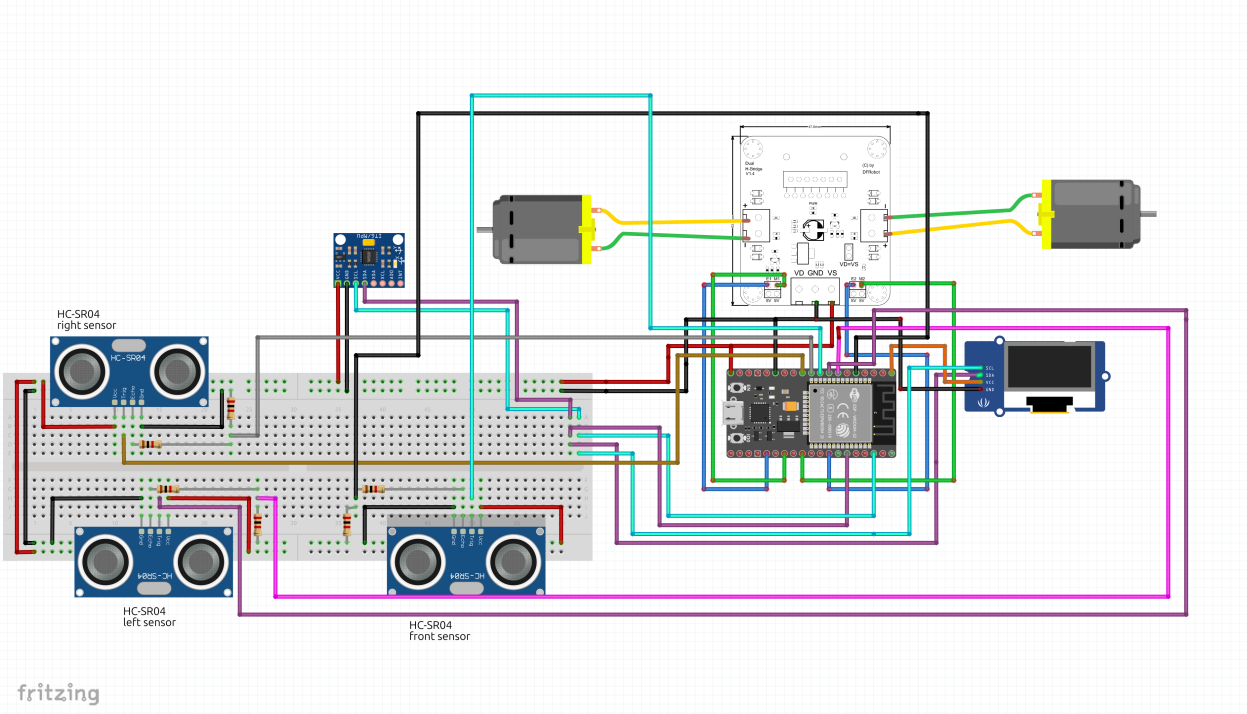


Figure 24: ESP32 Wiring

21.3.4. Software Configuration

1. **Arduino IDE Setup**: Install the Arduino IDE to program the ESP32 microcontroller. Ensure the ESP32 board is added to the Arduino IDE by following the instructions provided by Espressif Systems.
2. **Library Installation:** Install the ESP32_SSD1306 library to support the OLED display functionality.
3. **Code Upload:** Transfer the scripts located in the esp32 folder to the ESP32 device. Modify the WiFi settings in the script to match your local network configuration for connectivity.

21.4. Web Application Setup

21.4.1. Note To ensure a seamless setup of the virtual display, it is recommended to execute `docker-compose down` following each session.

21.4.2. Steps

1. The web application's source code is stored within the web app directory. Access this directory:

   ```
   cd ./web_app/
   ```

2. To launch the Docker containers, use the following commands:

   ```
   docker-compose up -d
   ```

21.5. Usage Instructions

1. Access the web application by navigating to http://localhost:8500 or http://localhost:5000 on your web browser.
2. Enter the ESP32's IP address within the web app and select the desired model for deployment.
3. The system provides an option for a virtual demonstration, allowing for operation without engaging the physical vehicle.
4. Initiate the maze navigation by clicking the `Start Maze` button.

A demonstration of the project is available (see Web App Demo in the Video References section).

21.6. Additional Information: Model Training

- Opt between utilizing a pre-trained model or conducting new training sessions using the script available in train.
- This training script is optimized for resource efficiency and can be executed directly on the Raspberry Pi.
- Upon completion, you will be prompted to save the new model. If saved, it will be stored within the models directory of the `web_app` folder.

Chapter 22. Video References

1. **Video E1 - Gyroscope Calibration**: Testing the MPU6050 gyroscope's ability to correct the car's orientation for accurate navigation, aiming to refine control over the vehicle's movement through maze environments.
    - Click here to go to the video: Video E1

Figure 25: QR code for video Video E1. (Video by author.)

2. **Video E2 - Navigational Corrections**: Addressing alignment issues when attempting precise 90-degree turns and realigning the car's forward movement to rectify a persistent ~3-degree offset.
    - Video E2

Figure 26: QR code for video Video E2. (Video by author.)

3. **Video E6 - Encoder Implementation**: Introducing rotary encoders to the setup, hoping to gain more precise control over the car's movements by accurately measuring wheel rotations, thus refining the vehicle's navigation capabilities.
    - Click here to go to the video: Video E6

Figure 27: QR code for video Video E6. (Video by author.)

4. **Video E7 - Troubleshooting Encoder Malfunction**: Addressing a malfunction with one of the encoders that halted further tests, highlighting the practical challenges of maintaining hardware reliability.
    - Click here to go to the video: Video E7

Figure 28: QR code for video Video E7. (Video by author.)

5. **Video E9 - Outdoor Navigation Test**: Navigating the RC-car on uneven outdoor surfaces, where variations greatly affected performance, underscoring the importance of environmental factors in autonomous navigation.
    - Click here to go to the video: Video E9

Figure 29: QR code for video Video E9. (Video by author.)

6. **Video E11 - Indoor Controlled Test**: Conducting controlled indoor tests to closely monitor and adjust the RC-car's navigation strategies, reflecting on the complexities of sim-to-real transfer.
    - Click here to go to the video: Video E11

Figure 30: QR code for video Video E11. (Video by author.)

7. **Web App Demo**: A demonstration of the web application's functionality, showcasing the user interface and the autonomous navigation system's control features.
    - Click here to go to the video: Web App Demo

Figure 31: QR code for video Web App Demo. (Video by author.)

8. **DDQN Simulation test**: A simulation test of the DDQN model navigating a maze environment, demonstrating the model's learning capabilities and decision-making processes.
    - Click here to go to the video: DDQN Simulation

Figure 32: QR code for video DDQN Simulation. (Video by author.)

References

[1] G. Brockman et al., “OpenAI Gym,” arXiv preprint arXiv:1606.01540, 2016.

[2] A. Dosovitskiy et al., “CARLA: An Open Urban Driving Simulator,” in Proceedings of the 1st Annual Conference on Robot Learning, 2017.

[3] H. Van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2016.

[4] J. Schulman et al., “Proximal Policy Optimization Algorithms,” arXiv preprint arXiv:1707.06347, 2017.

[5] J. Tobin et al., “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World,” in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[6] K. Bousmalis et al., “Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping,” in IEEE International Conference on Robotics and Automation (ICRA), 2018.

[7] Y. Pan and Q. Yang, “A Survey on Transfer Learning,” IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.

[8] A. A. Rusu et al., “Sim-to-Real Robot Learning from Pixels with Progressive Nets,” in Proceedings of the Conference on Robot Learning, 2016.

[9] S. James et al., “Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks,” in Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), 2019.

[10] F. Sadeghi and S. Levine, “(CAD)^2RL: Real Single-Image Flight without a Single Real Image,” in Proceedings of Robotics: Science and Systems, 2016.

[11] “Self Driving and Drifting RC Car using Reinforcement Learning,” YouTube, Aug. 19, 2019. [Online Video]. Available: https://www.youtube.com/watch?v=U0-Jswwf0hw. [Accessed: Jan. 29, 2024].

[12] Q. Song et al., “Autonomous Driving Decision Control Based on Improved Proximal Policy Optimization Algorithm,” Applied Sciences, vol. 13, no. 11, Art. no. 11, Jan. 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/11/6400. [Accessed: Jan. 29, 2024].

[13] DailyL, “Sim2Real_autonomous_vehicle,” GitHub repository, Nov. 14, 2023. [Online]. Available: https://github.com/DailyL/Sim2Real_autonomous_vehicle. [Accessed: Jan. 29, 2024].

[14] “OpenGL inside Docker containers, this is how I did it,” Reddit, r/docker. [Online]. Available: https://www.reddit.com/r/docker/comments/8d3qox/opengl_inside_docker_containers_this_is_how_i_did/. [Accessed: Jan. 29, 2024].

[15] M. A. Dharmasiri, “Micromouse from scratch | Algorithm- Maze traversal | Shortest path | Floodfill,” Medium, [Online]. Available: https://medium.com/@minikiraniamayadharmasiri/micromouse-from-scratch-algorithm-maze-traversal-shortest-path-floodfill-741242e8510. [Accessed: Jan. 29, 2024].

[16] “Reinforcement Learning with Multi-Fidelity Simulators – RC Car,” YouTube, Dec. 30, 2014. [Online Video]. Available: https://www.youtube.com/watch?v=c_d0Is3bxXA. [Accessed: Jan. 29, 2024].

[17] W. Zhao, J. P. Queralta, and T. Westerlund, “Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey,” in 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Dec. 2020, pp. 737–744. [Online]. Available: https://arxiv.org/pdf/2009.13303.pdf.

[18] R. S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: The MIT Press, 2018.

[19] H. van Hasselt, A. Guez, D. Silver, et al., “Deep Reinforcement Learning with Double Q-learning,” arXiv preprint arXiv:1509.06461, 2015.

[20] Papers With Code, “Double DQN Explained,” [Online]. Available: https://paperswithcode.com/method/double-dqn.

[21] D. Jayakody, “Double Deep Q-Networks (DDQN) - A Quick Intro (with Code),” 2020. [Online]. Available: https://dilithjay.com/blog/2020/04/18/double-deep-q-networks-ddqn-a-quick-intro-with-code/.

[22] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proc. of AAAI Conf. on Artificial Intelligence*, 2016.

[23] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.

[24] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[26] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Proc. of the 13th International Conf. on Neural Information Processing Systems*, pp. 1008-1014, 2000.

[27] T. Saanum, “Reinforcement Learning with Simple Sequence Priors,” arXiv preprint arXiv:2305.17109, 2024.

[28] “COMP 542: Machine Learning Cite Your Sources,” CSUN, [Online]. Available: https://libguides.csun.edu/comp542/cit
[Accessed: 12-May-2024].

[29] “IEEE Editorial Style Manual (Online),” IEEE, 2023. [Online]. Available: https://www.ieee.org/content/dam/ieee-org/ieee/web/org/conferences/style_references_manual.pdf. [Accessed: 12-May-2024]