Exploring the Feasibility of Sim2Real Transfer in Reinforcement Learning

Lucas Driessens

Bachelor Multimedia and Creative Technologies

University of Applied Sciences

Howest Kortrijk

2022

Abstract

In this research project, I delve into the fascinating realm of artificial intelligence, specifically focusing on reinforcement learning (RL) and its application in real-world scenarios. The crux of my investigation revolves around the challenging question 'Is it possible to transfer a trained RL agent from a simulation to the real world?' This inquiry is particularly examined in the context of maze navigation. This research is partitioned into sub-questions, which collectively aim to create a comprehensive understanding of the process. Firstly, I explore the various virtual environments available for training a virtual RF-car, seeking the most effective platform for my purposes. Secondly, I delve into identifying the most suitable reinforcement learning techniques for this specific application, considering factors like efficiency, adaptability, and real-world applicability. Lastly, the research seeks to bridge the gap between simulation and reality, investigating the practicality and challenges involved in this transition. Through this study, I aspire to contribute significantly to the field of AI and robotics, offering insights and methodologies that could potentially advance the implementation of RL in real-world applications. The outcomes of this research could have far-reaching implications, not only in robotics but also in areas where simulation-based training is crucial.

Table of Contents

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.

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

Introduction

In the evolving landscape of artificial intelligence and robotics, the distinction between virtual simulations and real-world applications increasingly narrows, presenting unprecedented opportunities and challenges. This thesis explores the potential of Reinforcement Learning (RL) to bridge this gap, with a specific focus on the domain of autonomous navigation using a remote-controlled (RC) car in a maze. The endeavor to transfer a trained RL agent from a simulated environment to the real world encapsulates the core challenge of sim-to-real transferability, a pivotal step towards realizing the full spectrum of RL's applicability in complex, real-world scenarios.

The purpose of this study is to explore the feasibility and challenges of transferring a trained RL agent from a simulated environment to the real world. This transition, known as "sim2real," is particularly examined in the context of maze navigation using a remote-controlled (RC) car. The significance of this research lies in its potential to bridge the gap between theoretical RL models and practical, real-world applications, which is a critical step in advancing the field of AI and robotics.

Background on Reinforcement Learning (RL)

Reinforcement Learning is a paradigm where agents learn to make decisions through trial and error, interacting with their environment to maximize cumulative rewards. Central to RL are the concepts of agents, environments, actions, states, and rewards, governed by Markov Decision Processes (MDP):

S is a set of states

A is a set of actions

$P(s*t+1|s_t, a_t)$ is the probability that action a_t in state s_t at time t will lead to state $s*t+1$

$R(s*t, a_t)$ is the reward received after transitioning from state s_t to state $s*t+1,$ due to action a_t

RL's versatility is showcased in its applications across various sectors, including autonomous vehicles, where it promises to enhance navigation, decision-making, and real-time adaptation.

Research Questions

This investigation is anchored by the question: "Can a trained RL agent be effectively transferred from a simulation to a real-world environment for maze navigation?" Addressing this question involves exploring multiple facets of RL training and implementation:

1. Selection of virtual environments for effective RL training.
2. Identification of RL techniques suited for autonomous navigation.
3. Evaluation of sim-to-real transfer in adapting to real-world dynamics.
4. Assessment of training efficacy and performance optimization through simulation.
5. Adaptation and transfer of a trained model to a real RC car, including necessary adjustments for real-world application.

A blend of qualitative and quantitative research methods, including simulation experiments, real-world trials, and literature review, form the methodological backbone of this study. This comprehensive approach aims to validate the sim-to-real transfer while contributing to the broader discourse on RL's practical applications and challenges.

Main Research Question

Is it possible to transfer a trained RL-agent from a simulation to the real world? (case: maze)

Sub Research Questions

1. Which virtual environments exist to train a virtual RC-car?
2. Which reinforcement learning techniques can I best use in this application?
3. Can the simulation be transferred to the real world? Explore the difference between how the car moves in the simulation and in the real world.
4. Does the simulation have any useful contributions? In terms of training time or performance?
5. How can the trained model be transferred to the real RC car? (sim2real) How do you need to adjust the agent and the environment for it to translate to the real world?

Methodology

This section explores the Reinforcement Learning Maze Navigation (RCMazeEnv) method, utilizing a Double Deep Q-Network (DDQNAgent) architecture. It details the maze environment setup, the DDQN agent design, and the comprehensive training algorithm, incorporating mathematical functions to delineate the system's mechanics.

Environment Setup (RCMazeEnv) The RCMazeEnv, a custom maze navigation environment derived from the OpenAI Gym framework, is designed for a 12x12 cell grid maze navigation task. Each cell within this grid can be identified as either a wall, represented by '1', or a path, represented by '0', with the goal designated at cell position (10, 10). The agent, visualized as a car, commences its journey from the starting position at cell (1, 1), facing eastward initially. The agent's navigation capabilities are enabled through a set of possible actions: moving forward, turning left, and turning right.

To assist in navigation, the agent is equipped with sensors that provide readings in three directions: front, left, and right. These sensors measure the distance to the nearest wall in their respective directions, offering crucial environmental information that aids in decision-making. The environment's state space, denoted as $\mathcal{S}$, encapsulates the agent's current position (x, y), its orientation θ, which can be one of $\{N, E, S, W\}$ representing north, east, south, and west respectively, and the sensor readings $\{s_{\text{front}}, s_{\text{left}}, s_{\text{right}}\}$. The goal of the agent is to navigate through the maze, from its starting point to the goal location, efficiently while avoiding collisions with walls and optimizing the path taken based on the sensor inputs and past experiences.

Agent Design (DDQNAgent) The agent employs a Double Deep Q-Network (DDQN) architecture to learn the optimal policy π^*. This is an enhancement over the standard DQN that aims to reduce overestimation of Q-values by decoupling the action selection from its evaluation:

- **Policy Network:** Estimates the Q-value $Q(s, a; \theta)$ for taking action a in state s, parameterized by weights θ.
- **Target Network:** Independently parameterized by weights θ^-, used to estimate the target Q-value for updating the policy network. It mirrors the architecture of the policy network but is updated less frequently to provide stable target values.

The Q-function update equation in DDQN is modified to:

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

Where:

$$R_{t+1}\, is the reward received after taking action a\, in state s$$

$$\gamma\, is the discount factor.$$

$$\underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \theta)\, selects the action using the policy network.$$

$$Q\left(S_{t+1}, a; \theta^-\right)\, evaluates the action using the target network.$$

The action space $\mathcal{A}$ and the rest of the agent's setup remain as previously described. The DDQN architecture significantly improves the stability and performance of the agent by addressing the overestimation of Q-values, promoting a more accurate and reliable learning process.

Training Process The training process utilizes the experience replay mechanism, storing transitions (s, a, r, s') in a replay buffer D. The DQN is trained by minimizing the loss function $L(\theta)$ defined as the mean squared error 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 θ^- represents the weights of a target network, and γ is the discount factor. The target network's weights are periodically updated to match the policy network, stabilizing training.

The epsilon-greedy strategy is employed for action selection, with ϵ gradually decaying from 1 to a minimum value, balancing exploration and exploitation.

Reward Function Components

Collision Penalty $R_{\text{collision}}$ When the agent attempts to move into a wall or outside the designated maze boundaries, it triggers a collision state. To discourage such actions, which are counterproductive

to the goal of reaching the destination, a significant penalty is applied. This penalty is critical for teaching the agent about the boundaries and obstacles within the environment, ensuring that it learns to navigate safely and effectively.

$$R_{\text{collision}} = -20$$

Goal Achievement Bonus R_{goal} Reaching the goal is the primary objective of the maze navigation task. A substantial reward is given to the agent upon achieving this objective, signifying the completion of the episode. This reward serves as a strong positive reinforcement, guiding the agent's learning towards the goal-oriented behavior. However, an additional mechanism penalizes the agent if it takes an excessively long route to reach the goal, promoting efficiency in navigation.

$$R_{\text{goal}} = \begin{cases} +500, & \text{if goal is reached} \\ -200, & \text{if steps} > 1000 \end{cases}$$

Proximity Reward $R_{\text{proximity}}$ This component of the reward function incentivizes the agent to minimize its distance to the goal over time. By rewarding the agent based on its proximity to the goal, it encourages exploration and path optimization, guiding the agent to navigate the maze more effectively. The reward decreases as the distance to the goal increases, encouraging the agent to always move towards the goal.

$$R_{\text{proximity}} = \frac{50}{d_{\text{goal}} + 1}$$

Progress Reward R_{progress} The progress reward or penalty is designed to encourage the agent to make decisions that bring it closer to the goal and to penalize decisions that lead it away. This dynamic reward system provides immediate feedback based on the agent's movement relative to the goal, promoting smarter navigation decisions.

$$R_{\text{progress}} = \begin{cases} +50, & \text{if distance decreases} \\ -25, & \text{if distance increases} \end{cases}$$

Exploration Penalty R_{revisit} To discourage repetitive exploration of the same areas, which indicates inefficient pathfinding, the agent receives a penalty for re-entering previously visited cells. This penalty is crucial for encouraging the exploration of new paths and preventing the agent from getting stuck in loops or dead ends.

$$R_{\text{revisit}} = -10$$

Efficiency Penalty $R_{\text{efficiency}}$ Every step the agent takes incurs a small penalty. This mechanism ensures that the agent is incentivized to find the shortest possible path to the goal, balancing the need to explore the environment with the goal of reaching the destination as efficiently as possible.

$$R_{\text{efficiency}} = -2$$

Evaluation and Termination Conditions The reward function $R(s, a)$ is designed to encourage reaching the goal while penalizing collisions and inefficient paths. The reward for each step is defined as:

$$R(s,a) = \begin{cases} 500 & \text{if goal is reached} \\ -20 & \text{if collision} \\ 50/(d+1) & \text{otherwise} \end{cases}$$

where d is the Euclidean distance to the goal, encouraging the agent to minimize the distance to the goal.

The episode terminates when the agent reaches the goal, collides with an obstacle, or exceeds a predefined step limit, aiming to learn an efficient navigation policy.

Experimental Outcomes and Implementation Details

The project embarked on a journey to bridge the virtual and real-world through a meticulously designed environment and a cutting-edge agent architecture.

Virtual Environment and Agent Design

- **RCMazeEnv**: Customized for this project, the environment simulates a robotic car navigating a maze. Its design replicates real-world physics and constraints, offering a rich testing ground for reinforcement learning algorithms. The maze's structure, from its starting position to the goal, and the robotic car's specifications, including movement actions and sensor setups, are critical to the simulation's realism.
- **Double Deep Q-Network (DDQN)**: Employing two neural networks, this model enhances traditional reinforcement learning methods by reducing the overestimation of Q-values. The policy

network and the target network work in tandem to refine the agent's learning process through continuous interaction and sensor data interpretation.

Implementation Highlights

- **Environment and Agent Interaction**: Central to the DDQN agent's strategy is its continuous adaptation to the environment, leveraging sensor inputs to inform its decisions and optimize its path through the maze. This iterative learning process is visually represented through a simulation platform that allows for detailed observation of the agent's performance and strategy adjustments.
- **Real-World Application**: Transferring the virtual training to a physical RC robot involved comprehensive hardware setup and calibration. Challenges such as sensor data normalization and precise movement control were addressed to ensure a seamless transition from virtual to real-world application.

Evaluation and Metrics The project employed specific metrics to evaluate the agent's efficiency in navigating the maze, with emphasis on both simulation performance and real-world applicability. This involved monitoring the agent's episodic performance, step efficiency, and adaptation to real-world conditions.

Unique Features

- **Physical Maze and Web Application**: A constructed physical maze served as the tangible counterpart to the virtual `RCMazeEnv`, playing a crucial role in testing the RC robot's navigation capabilities. Additionally, a web application was developed to act as a visualization and control interface, enhancing the interaction between the virtual and real-world applications.

Model Architecture and Training Insights

The Double DQN model's architecture is central to understanding the agent's learning and decision-making capabilities. Structured with four dense layers, it outputs three actions tailored to the RC car's movement, enabling sophisticated navigation strategies within the maze.

Model Architecture:

```
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)
---
```

This model is instrumental in the agent's ability to learn from its environment, adapting its strategy to optimize for both efficiency and effectiveness in maze navigation.

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`)**: 0.99993
  - 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.

Training Procedure

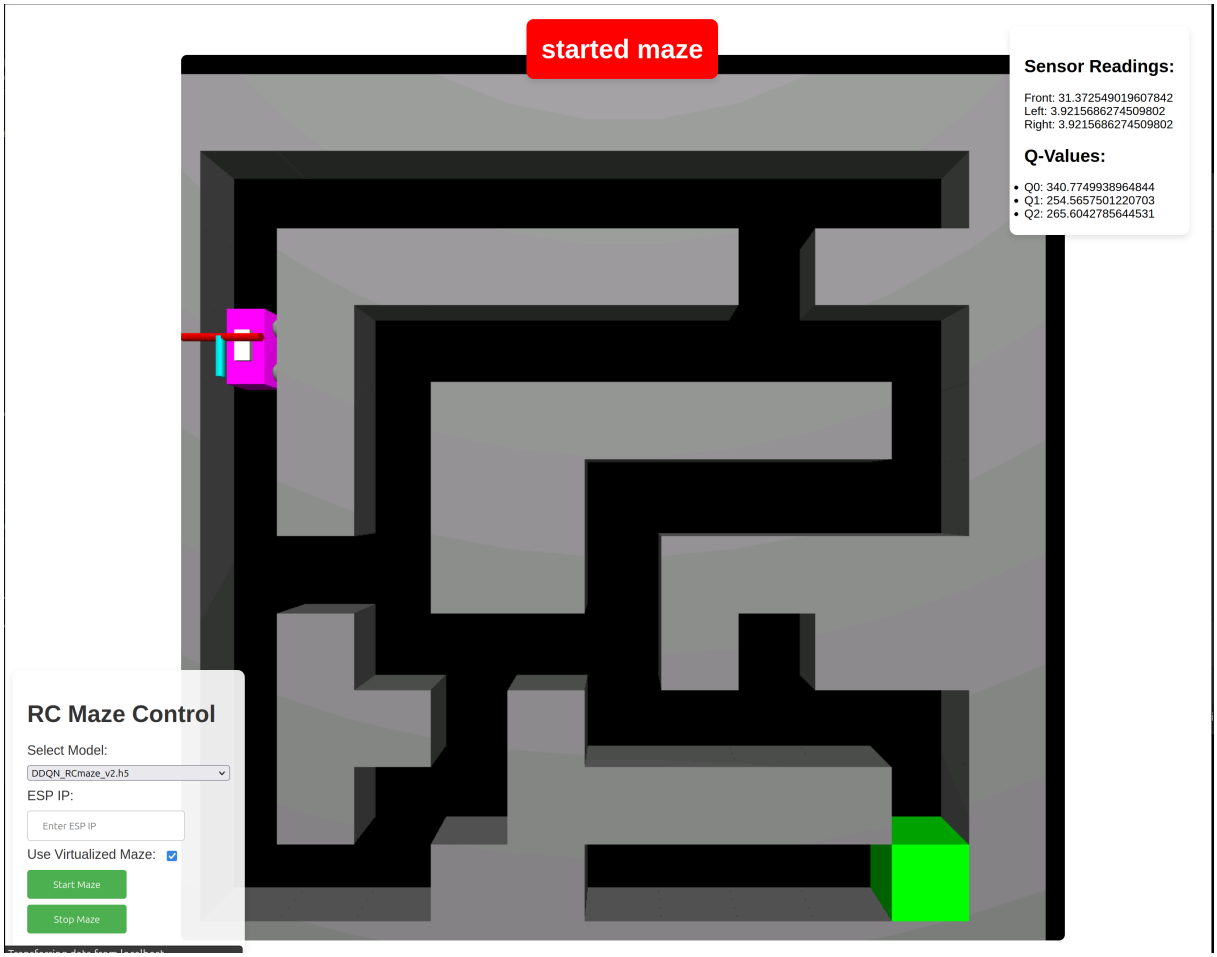
1. **Initialization**: Start with a high exploration rate (`EPSILON`) allowing the agent to explore the environment extensively.
2. **Episodic Training**: For each episode, the agent interacts with the environment, collecting state, action, reward, and next state data.
3. **Replay Buffer**: Store these 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.

Visual Insights and Further Exploration

The project's innovative approach to sim-to-real transfer in reinforcement learning is encapsulated in a series of visual representations and demonstrations, from the detailed construction of the physical maze to the dynamic interface of the web application.

- **Maze Visualization:**

- **Web Application Interface:**



- **Simulation Test Video:**

DDQN Test in Action

Evaluation Metrics Overview

Evaluating the performance of the Double Deep Q-Network (DDQN) agent both in simulation and real-world scenarios was essential to assess the effectiveness of reinforcement learning strategies applied to maze navigation. This evaluation provides insights into the agent's learning progress, its decision-making efficiency, and the challenges faced when transitioning from a virtual to a tangible maze.

Simulation Metrics The primary objective within the simulated environment was to determine the agent's capability to solve the maze with optimal efficiency and minimal errors.

- **Episodic Performance**: By analyzing the number of episodes required for consistent maze resolution, insights into the learning curve and adaptation of the agent were gained. Consistent maze resolution with fewer episodes indicates effective learning and strategy optimization.
- **Step Efficiency**: The efficiency with which the agent completes the maze—measured in steps—sheds light on its decision-making process and path optimization capabilities. Fewer steps to reach the goal suggest a higher level of learning and efficiency.
- **MSE Loss Measurement**: The mean squared error (MSE) formula quantifies the difference between the predicted values by the agent and the actual values, providing a mathematical measure of the agent's prediction accuracy.

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

- **Reward Trend Analysis**: Monitoring the reward history offers an understanding of how the agent's actions lead to positive or negative outcomes over time, illustrating the agent's growing proficiency in navigating the maze.
- **Epsilon Decay Tracking**: The adjustment of the epsilon value—balancing exploration and exploitation—over episodes is crucial for fostering an optimal learning pace. This metric demonstrates the agent's transition from exploring the maze to exploiting known paths for success.

Real-World Metrics Transitioning to real-world application involved assessing how the simulation-trained agent's strategies fared in a physical maze with tangible obstacles and limitations.

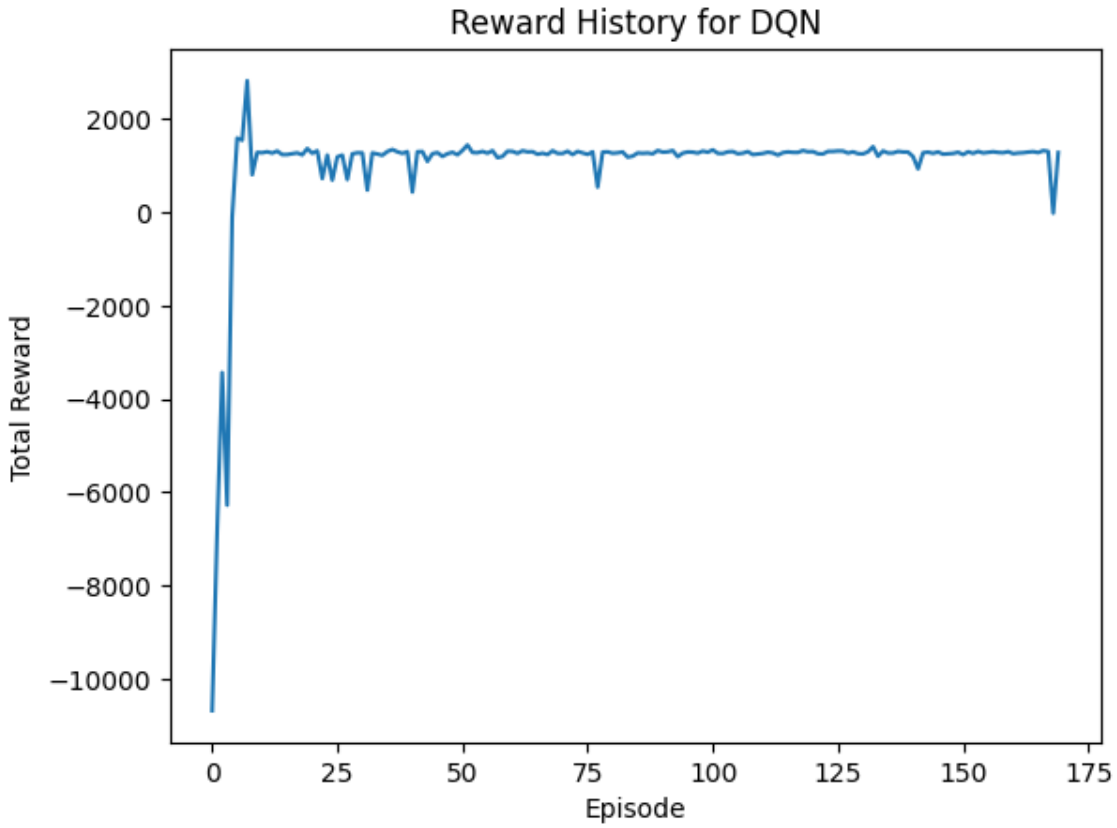
- **Maze Navigation**: A visual assessment of the RC car's ability to navigate a real-world maze provided direct evidence of the sim-to-real transfer's effectiveness, highlighting the practical application of the trained agent.
- **Sensor Data Analysis**: Evaluating the real-time sensor data in navigation scenarios enabled a detailed understanding of the agent's interaction with the physical world, particularly in terms of collision avoidance and pathfinding efficiency.

results

Reinforcement Learning Techniques Overview

1. Deep Q-Network (DQN)

- **Description**: The Deep Q-Network (DQN) combines a deep neural network with a Q-learning framework. It excels in handling high-dimensional sensory inputs, making it ideal for environments demanding detailed interaction.
- **Suitability**: DQN's advanced learning capabilities are tempered by its tendency to overestimate Q-values in complex environments. This limitation could affect its effectiveness in training RC-cars, where environmental dynamics are unpredictable.
- **Integration and Results**:
  - **Reward History**:

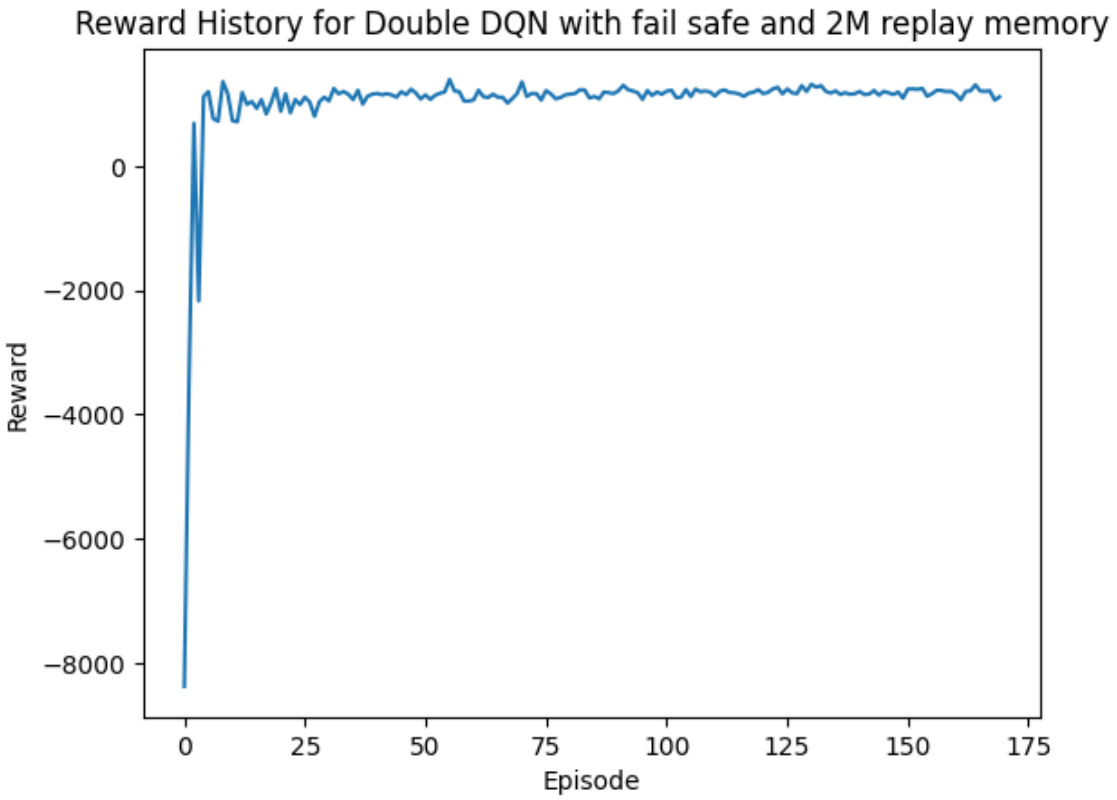


  - **Performance**: DQN's performance, while competent, was limited by Q-value overestimation in intricate scenarios.

2. Double Deep Q-Network (DDQN)

- **Description**: The Double Deep Q-Network (DDQN) improves upon DQN by employing two neural networks. This structure effectively reduces overestimation bias by separating action selection from Q-value generation.

- **Reason for Selection**:
  - DDQN's accuracy in Q-value approximation is crucial for navigating complex environments, such as mazes.
  - The RC-car's sensor limitations, which could lead to Q-value overestimations, are better addressed by DDQN.
  - Empirical trials showed DDQN's superior performance in maze navigation tasks.
- **Integration and Results**:
  - **Reward History**:

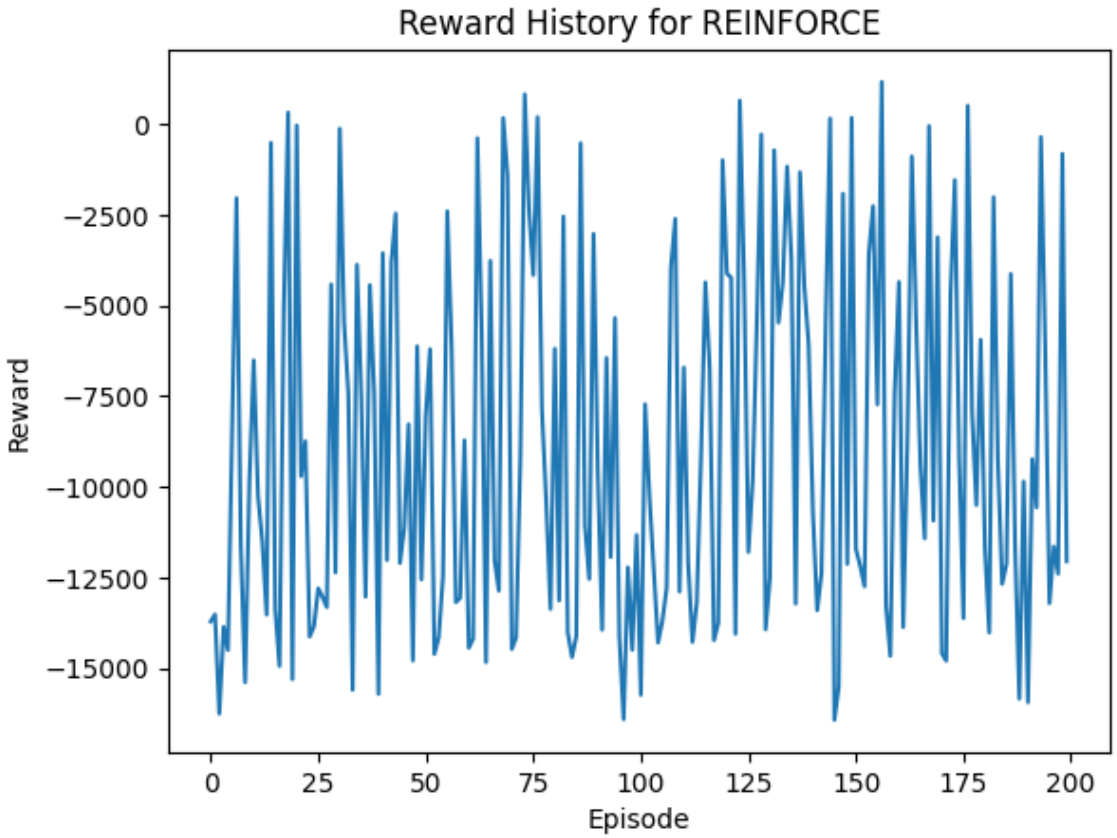


  - **Performance**: DDQN solved the environment in an average of 25 steps, compared to DQN's 34 steps, highlighting its efficiency.

3. Proximal Policy Optimization (PPO)

- **Description**: Proximal Policy Optimization (PPO) is a policy gradient method that directly optimizes decision-making policies. It's known for its stability and efficiency in specific RL contexts.

- **Suitability**: PPO's emphasis on policy optimization over value estimation makes it less suitable for RC-car simulations, where accurate Q-value approximation is key.
- **Integration and Results**:
  - **Reward History**:

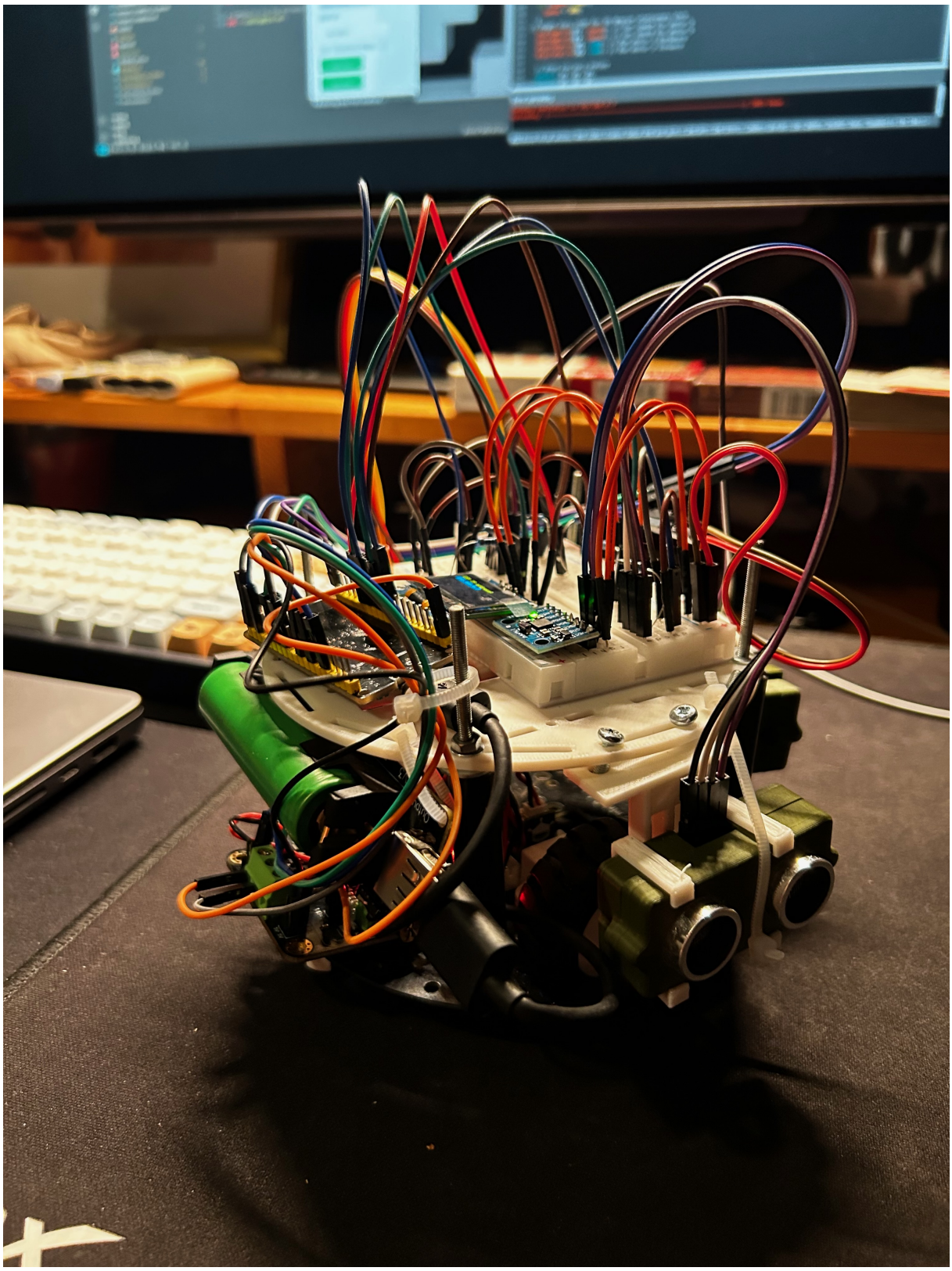


  - **Performance**: PPO, while stable, did not align well with the precision requirements for RC-car maze navigation.

Hardware Setup and Assembly

Introduction to Hardware Components

This section provides a detailed overview of the hardware components used in the research project, focusing on the assembly and configuration of the RC robot designed for maze navigation.

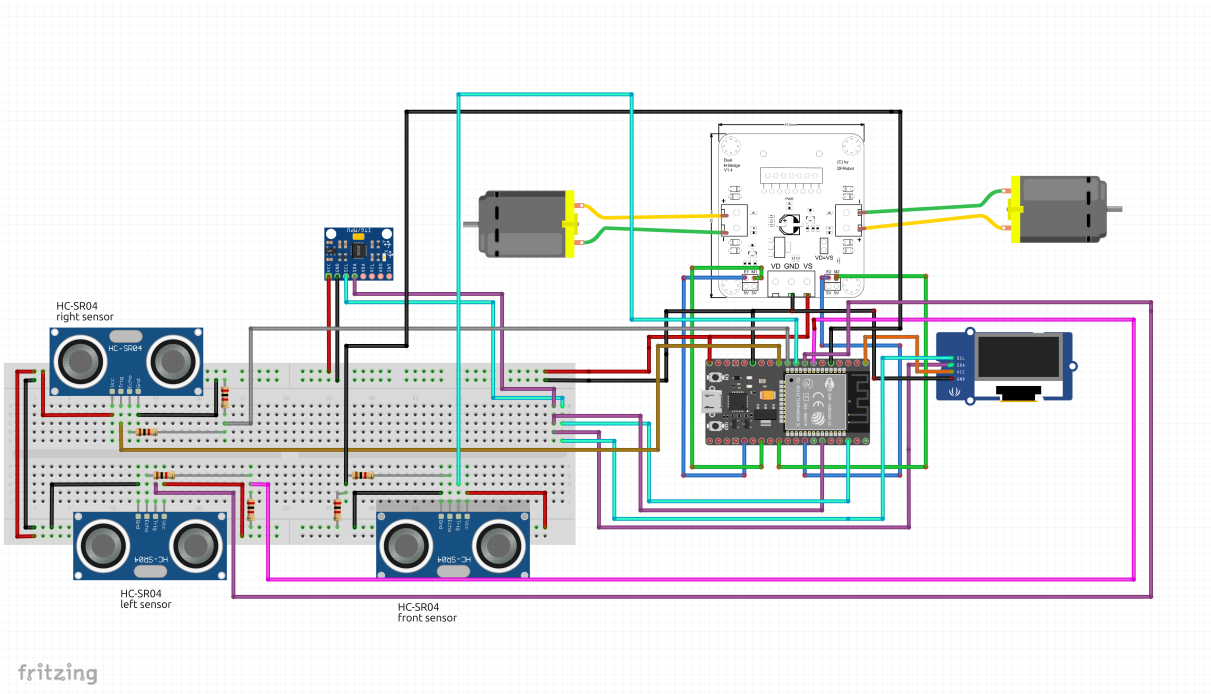


Components List

- **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

Wiring Guide

ESP32 Wiring::



Challenges and Solutions in Implementing RL Techniques and Virtual Environments

Challenge 1: Selection of an Appropriate Virtual Environment

- **Description**: Choosing a virtual environment conducive to effective RC-car training is crucial.
- **Solution**: After evaluating various platforms, **OpenAI Gym** was selected for its simplicity, familiarity from previous coursework, and its focus on reinforcement learning.

Challenge 2: Choosing the Optimal Reinforcement Learning Technique

- **Description**: Selecting the most effective RL technique for training the virtual RC-car.
- **Solution**: Through comparative analysis and empirical testing, the Double Deep Q-Network (DDQN) was identified as the most suitable technique, demonstrating superior performance in navigating complex environments with fewer episodes.

Challenge 3: Sim2Real Transfer - Addressing Movement Discrepancies

- **Description**: Bridging the gap between simulation and real-world in terms of RC-car movement and control.
- **Solution Attempt**: Fine-tuning the frequency of action commands with an async method, waiting for the motor to finish moving or considering a queued action system. Further more the importance of precise movement in the real world was highlighted, which was not a problem in the simulation.

Challenge 4: alignment Issue and Motor Encoder Implementation

- **Description**: Difficulty in achieving precise straight-line movement in the RC car, with a persistent ~3-degree offset.
- **Solution Attempt 1**: Implementation of motor encoders was pursued to enhance movement accuracy. However, this approach faced the same limitations in achieving the desired precision.
- **Solution Attempt 2**: The motor was replaced with a more powerful one, which initially showed promise in addressing the alignment issue. However, after adding all the other components, the car's weight increased, leading to the same problem.
- **Solution Attempt 3**: The use of a MPU6050 gyroscope was explored to measure the car's orientation and adjust the movement accordingly. Even though this approach succeeded to some extent (90 degrees turns were accurate), it was not able to solve the ~3-degree offset issue when moving forward.

- **Solution Attempt 4**: The final solution I tried was done by removing the RPI5 (previously used for sensor data and running the web app) from the robot all together and using the ESP32 to control both all the sensors and the motors. This allowed for a more lightweight robot, which was able to move forward more precisely but it failed to rotate 90 degrees accurately.

Challenge 5: Ensuring Consistent and Effective Training

- **Description**: Maximizing training efficiency and performance while maintaining consistency between simulation and real-world scenarios.
- **Solution**: The simulation demonstrated considerable advantages in terms of training efficiency, safety, and computational power, establishing it as an indispensable tool in autonomous vehicle model development.

Challenge 6: Accurate Sensor Data Normalization for Sim2Real Transfer

- **Description**: Aligning sensor data between simulated and real-world environments is critical for model accuracy.
- **Solution**: Implementing specific normalization techniques for both real-world and simulation sensor data ensured consistency and compatibility, enhancing the model's accuracy in real-world applications.
  - **Real-World Sensor Data Normalization:**

    The function `map_distance` normalizes real-world sensor data. It can be represented as follows:

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

    This function keeps distances under 25 cm unchanged and applies a scaling factor of 0.5 to distances beyond 25 cm, adding this scaled value to a base of 25 cm.

  - **Simulation Sensor Data Normalization:**

    The function `normalize_distance` adjusts simulated sensor data to a 0-1 range. Its equation is:

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

In this function, the distance is first scaled by dividing by `sensor_max_range`. It's then clamped between 0 and 1 before multiplying by 1000 to normalize it within a specific range.

Challenge 7: Integration of Failsafe Mechanisms

- **Description**: Preventing potential collisions and ensuring safe navigation in the real world.
- **Solution**: Development of a failsafe system that prevents forward movement in hazardous situations, retraining the model with this protocol to align real-world behavior with the simulated environment.

Challenge 8: Training Environment and Technique Efficacy

- **Description**: Determining the most effective environment and RL technique for training.
- **Solution**: The DDQN solved the environment more efficiently than DQN, highlighting the importance of technique selection. The simulation provided a safer, more controlled environment for training, reinforcing its selection over real-world training.

Viewing Practical Experiments

For visual insights into my practical experiments addressing these challenges, please refer to my supplementary video materials, which illustrate the implementation and testing of solutions, from gyroscopic adjustments to the integration of a more sophisticated control system using the ESP32.

Conclusion

This section has outlined the practical challenges encountered in applying reinforcement learning (RL) techniques to autonomous RC cars. My journey began with the selection of OpenAI Gym as the virtual environment, chosen for 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 the implementation of 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 replacing the Raspberry Pi and ESP32 with just the ESP32 module in the robot's design, leading to a more lightweight and precise robot. This change marked a considerable step in overcoming the challenges previously faced.

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

In conclusion, this project underscores the iterative and demanding nature of applying RL techniques in real-world scenarios. It highlights the importance of continuous refinement, innovation, and adaptation, beyond the theoretical knowledge base. The journey through these challenges has emphasized the significance of perseverance and creative problem-solving in the evolving field of autonomous vehicle technology.

Real-World Application and Limitations

Introduction to Sensor and Movement Discrepancies

The leap from simulated environments to real-world application unveils a complex landscape of challenges, especially in the interpretation of sensor data and the replication of vehicle movements. This discussion delves into these critical aspects, highlighting both the opportunities and constraints of applying simulation-derived insights to actual autonomous vehicle (AV) operations.

Real-World Application

Enhanced Sensor-Based Navigation Sensor-based navigation technologies, refined through simulation, promise substantial improvements in autonomous vehicles' functionality. In real-world applications, such technologies are pivotal for environments demanding high precision and adaptability. For instance, in congested urban settings or in automated delivery systems, the ability to dynamically navigate with high accuracy can significantly elevate both safety and efficiency. Integrating simulation insights into sensor-based navigation aids in refining these systems to better interpret complex, variable real-world conditions.

Informing Autonomous Vehicle Movement Simulated environments offer 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, thereby improving the overall functionality of autonomous transportation.

Limitations

Discrepancies in Sensor Data Interpretation A substantial hurdle 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 and, by extension, its safety and operational efficiency.

Challenges in Movement Replication The precise replication of simulated vehicle movements in 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.

Practical Implementation Considerations Successfully translating simulation insights into real-world applications requires meticulous attention to several practical aspects. These include, but are not limited to, 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 sim2real insights.

Conclusion

Transitioning from simulation-based research to practical real-world applications in autonomous vehicle navigation presents a unique set of challenges and opportunities. While the application of 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.

Answers to Research Questions

1. Virtual Environments for RF-Car Training

The choice of a virtual environment is paramount in simulating the complex dynamics of autonomous driving. Platforms such as Unity 3D, AirSim, CARLA, OpenAI Gym, and ISAAC Gym offer varied features catering to different aspects of driving simulation. However, for RF-car training, OpenAI Gym is selected for its flexibility in custom environment creation and its compatibility with Python, facilitating ease of use and integration with existing advanced AI coursework [1].

Unity 3D and AirSim, while providing realistic simulations, require expertise beyond Python, limiting their accessibility for the current project scope. CARLA offers comprehensive autonomous driving simulation capabilities but is tailored towards more traditional vehicle models rather than RF-cars. ISAAC Gym, with its focus on robotics, presents a similar mismatch in application. In contrast, OpenAI Gym's simplicity and reinforcement learning focus make it an ideal platform for this project, supporting effective SIM2REAL transfer practices [2].

2. Reinforcement Learning Techniques for Virtual RF-Car Training

The comparison of Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and Proximal Policy Optimization (PPO) techniques reveals that DDQN offers the best fit for the project's needs. DDQN's architecture, designed to address the overestimation bias inherent in DQN, enhances accuracy in Q-value approximation—a critical factor in navigating the complex, sensor-driven environments of RF-car simulations [3].

DQN, while powerful for high-dimensional sensory input processing, falls short in environments with unpredictable dynamics, a limitation DDQN effectively overcomes. PPO's focus on direct policy optimization provides stability and efficiency but lacks the precision in value estimation necessary for RF-car training. Empirical trials further validate DDQN's superior performance, demonstrating its suitability for the intricate maze-like environments encountered by virtual RF-cars [4].

3. Sim-to-Real Transfer Challenges and Solutions

Transferring simulation models to real-world applications involves addressing discrepancies in sensor data interpretation, action synchronization, and physical dynamics. Solutions such as sensor data normalization and action synchronization mechanisms were implemented to align simulation outcomes with real-world performance [5].

The introduction of failsafe mechanisms and adjustments in motor control timings proved critical in mitigating issues like collision risks and movement inaccuracies, underscoring the importance of iterative testing and adaptation in sim-to-real transfer [6].

4. Contributions of Simulation in RF-Car Training

Simulation training offers distinct advantages in efficiency, safety, and computational resources. It enables uninterrupted and automated training sessions, eliminates the risks associated with real-world training, and leverages powerful computing resources to accelerate the training process [7].

The comparative analysis between simulation and real-world training outcomes highlights the practicality and effectiveness of simulation in developing autonomous driving models, making it an indispensable tool in the RF-car development process [8].

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

Applying a trained model to a physical RC car requires careful consideration of environment, agent, and model adjustments. Strategies for effective sim-to-real adaptation include fine-tuning sensor interpretations, implementing action synchronization measures, and adjusting physical dynamics to mirror those of the simulation [9].

This process ensures the successful application of simulation training to real-world scenarios, facilitating the development of robust and reliable autonomous driving systems [10].

Reflection

The path from conceptualizing a virtual RF-car training simulation to its real-world application traverses the rich terrain of integrating theoretical research with tangible, practical outcomes. Reflecting on feedback, along with the journey itself, unveils crucial insights into the research process, its achievements, and areas ripe for growth:

Strengths and Weaknesses

The project's resilience in adapting to unforeseen challenges stands out as a testament to the robustness and flexibility of the research approach. This adaptability is underscored by the ability to pivot in methodology when confronted with real-world complexities not mirrored in the simulation. However, an initial hesitancy to venture beyond familiar tools and methodologies highlighted a potential limitation in fully leveraging the breadth of available technologies and approaches. This reticence, perhaps

rooted in comfort with established practices, may have initially narrowed the scope of exploration and innovation.

Practical Applicability and Industry Relevance

The feedback collectively emphasizes the practical applicability and value of the project's findings within the industry. The methodology and outcomes provide a concrete framework for navigating the intricacies of sim-to-real transitions, crucial for the development of autonomous vehicle technologies. This relevance extends beyond theoretical interest, suggesting a solid foundation for application in real-world autonomous system development.

Encountered Alternatives and Flexibility

The encouragement to explore sophisticated simulation environments and alternative machine learning methodologies resonates with a broader industry and academic expectation for versatile, dynamic research approaches. This suggests a pivotal learning moment: the importance of maintaining flexibility in both tools and conceptual frameworks to ensure research remains responsive and relevant to evolving technological landscapes and real-world demands.

Anticipated Implementation Barriers

Identifying anticipated challenges in corporate implementation, such as the need for significant investment and the integration of novel findings into established workflows, offers a grounded perspective on the path to practical application. This awareness is instrumental in bridging the gap between research outcomes and their industry adoption, guiding future strategies to mitigate these barriers.

Societal Contributions and Broader Impacts

Reflecting on the societal and economic impacts of the project broadens its significance beyond immediate industry application. The potential for safer, more efficient autonomous vehicle technologies underscores a broader contribution to societal welfare and technological advancement. This perspective enriches the reflection, highlighting the project's role in contributing to a future where autonomous technologies enhance safety, efficiency, and environmental sustainability.

Lessons Learned and Forward Path

This reflective journey underscores several key lessons: the value of openness to new methodologies, the importance of bridging theory with practice through versatile research approaches, and the critical role of anticipatory thinking in addressing implementation barriers. Looking forward, these insights pave the way for a research ethos characterized by adaptability, responsiveness to industry needs, and a commitment to contributing to societal progress through technological innovation.

Advice for those Embarking on Similar Research Paths

1. **Flexibility in Choosing Simulation Environments**
    - Begin your research with an open mind regarding the choice of simulation environments. While familiarity and ease of use are important, they should not be the sole criteria. The initial selection of OpenAI Gym was based on previous coursework experience, but this choice later proved to be limiting in replicating real-world movements of the car. Exploring and testing multiple environments can provide a better understanding of their capabilities and limitations, ensuring a more robust preparation for real-world application challenges.
2. **Expectation Management and Preparedness for the Unexpected**
    - Anticipate and plan for unexpected challenges that arise when transitioning from a simulated to a real-world environment. The real world introduces complexities and variables that are difficult to simulate accurately. Being prepared to iterate on your model and adapt your approach in response to these challenges is crucial for success.
3. **The Importance of Not Being Overly Committed to a Single Solution**
    - Avoid becoming too attached to a specific solution or methodology. The research process should be dynamic, allowing for the exploration of alternative approaches and solutions. Being open to change, even late in the research process, can uncover more effective strategies and technologies. This adaptability is especially important in fields like autonomous vehicle development, where technological advancements occur rapidly.
4. **Detailed Attention to Sensor Data and Real-World Variables**
    - Precision in sensor data interpretation and calibration is paramount. Discrepancies between simulated and real-world sensor data can significantly impact the performance and reliability of autonomous systems. Ensuring that your simulation accurately reflects the nuances of real-world sensor data will enhance the validity of your model and the smoothness of the transition to real-world application.

5. **The Value of Interdisciplinary Collaboration**
    - Engage with experts across different fields to leverage a wide range of perspectives and expertise. Collaborations with specialists in robotics, mechanical engineering, computer science, and other relevant fields can provide insights into challenges and solutions that may not be apparent from a single disciplinary viewpoint. This collaborative approach can lead to innovative solutions that significantly improve the quality and applicability of your research.
6. **Openness to Community and Industry Feedback**
    - Actively seek and incorporate feedback from both the academic community and industry professionals. Insights from Sam and Wouter, for example, highlighted the strengths and weaknesses of the research project, offering valuable perspectives on its applicability and potential implementation challenges in the business world. Feedback can guide refinements in methodology and approach, enhancing the relevance and impact of your work.
7. **Consideration of Socio-Economic Impacts**
    - Reflect on the broader implications of your research, including its potential socio-economic benefits. Autonomous vehicle technologies can have significant societal impacts, from improving transportation safety to enhancing mobility and reducing environmental footprints. Research in this field should consider these broader outcomes, aiming to contribute positively to society and the economy.

General Conclusion

This research has made significant strides in understanding the feasibility and challenges of Sim2Real transfers in reinforcement learning. While substantial progress was achieved, the journey illuminated the vast landscape of challenges that lie in the nuanced discrepancies between virtual and physical realms. Future endeavors in this domain should continue to push the boundaries of what is possible, leveraging the lessons learned to further bridge the gap between simulation and reality. The potential applications of successfully transferring RL agents to the real world are vast, promising advancements in robotics, autonomous vehicles, and beyond.

Credits

I am immensely grateful to my coach and supervisor, Gevaert Wouter, for his guidance and clever insights that significantly shaped the course of this research project. In addition to his invaluable

assistance during the project, I would also like to extend my thanks for the enjoyable courses he delivered during my time at Howest.

Sources of Inspiration and Conceptual Framework

The genesis of 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.

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.

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.

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.

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.

Integration of Practical Experiments

Throughout this research project, I employed a series of practical experiments to navigate and overcome encountered challenges. These experiments, documented through video demonstrations, provide tangible insights into my problem-solving process.

Addressing Alignment and Orientation Challenges

One of the key challenges I faced was ensuring precise orientation and alignment of the RC-car during movement. To tackle this, I utilized the MPU6050 gyroscope, aiming to correct alignment issues and achieve accurate 90-degree turns.

- **Utilizing the MPU6050 Gyroscope for Precise Orientation**: My first set of experiments focused on leveraging the gyroscope to correct the car's orientation for accurate navigation. This approach was pivotal in my attempts to ensure the RC-car could navigate mazes with high precision.
    - To address alignment issues when attempting precise 90-degree turns, I explored the potential of the MPU6050 gyroscope to adjust the car's movement based on its orientation. This experiment aimed to refine my control over the vehicle's navigation through the maze (View Test 1, View Test 2).
    - Further testing focused on using the gyroscope for realigning the car's forward movement, aiming to rectify the persistent ~3-degree offset. Despite my efforts, completely eliminating this offset proved challenging, showcasing the complexities of simulating real-world physics (View Test 1, View Test 2, View Test 3).

Enhancing Movement Precision with Encoders

The pursuit of enhancing the RC-car's movement precision led us to experiment with rotary encoders. These devices were integrated to measure wheel rotations accurately, aiming to improve straight-line movements and correct the noted ~3-degree offset.

- **Experimenting with Rotary Encoders**: I introduced rotary encoders to my setup, hoping to gain more precise control over the car's movements by accurately measuring wheel rotations. This experiment represented a significant effort to refine the vehicle's navigation capabilities by ensuring more accurate movement and orientation.
    - Initial tests with a new RC-car model, equipped with an encoder and a more powerful motor, showed promise in addressing the forward movement precision. However, the addition of extra components increased the vehicle's weight, impacting its movement and reintroducing the alignment challenge (View Test 1, View Test 2).
    - Despite an encouraging start, a malfunction with one of the encoders halted further tests using this specific setup, highlighting the practical challenges of hardware reliability in real-world applications (View Test).

Real-World Application Tests

Moving beyond controlled environments, I conducted tests in both outdoor and indoor settings to evaluate the RC-car's performance in real-world conditions. These tests were crucial for assessing the practical application of my research findings.

- **Outdoor and Indoor Maze Tests**: Real-world testing scenarios presented unique challenges, such as varying surface textures and unpredictable environmental conditions, which significantly impacted the RC-car's navigation capabilities.
    - The outdoor test attempted to navigate the RC-car on uneven surfaces, where surface texture variations greatly affected its performance. This test underscored the importance of environmental factors in autonomous navigation (View Test 1, View Test 2).
    - Indoor testing provided a more controlled environment, allowing us to closely monitor and adjust the RC-car's navigation strategies. Despite the controlled conditions, these tests highlighted the challenge of accurately translating simulation models to real-world applications, reflecting on the complexities of sim-to-real transfer (View Test 1, View Test 2, View Test 3, View Test 4).

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," *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," *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,” *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,” *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,” *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[10] F. Sadeghi and S. Levine, “(CAD)ˆ2RL: Real Single-Image Flight without a Single Real Image,” *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: <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.