

Miniature Autonomous Vehicle Environment for Sim-to-Real Transfer in Reinforcement Learning

Stephan Pareigis^a, Daniel Riege and Tim Tiedemann

Department of Computer Science, HAW Hamburg, Berliner Tor 7, 20099 Hamburg, Germany

Keywords: Reinforcement Learning, Digital Twin, Autonomous Driving, Sim-to-Real Gap, Miniature Autonomy, Real-World Reinforcement Learning.

Abstract: An experimental setup and preliminary validation of a platform for sim-to-real transfer in reinforcement learning for autonomous driving is presented. The platform features a 1:87 scale miniature autonomous vehicle, the tinycar, within a detailed miniature world that includes urban and rural settings.

Key components include a simulation for training machine learning models, a digital twin with a tracking system using overhead cameras, an automatic repositioning mechanism of the miniature vehicle to reduce human intervention when training in the real-world, and an encoder based approach for reducing the state space dimension for the machine learning algorithms.

The tinycar is equipped with a steering servo, DC motor, front-facing camera, and a custom PCB with an ESP32 micro-controller. A custom UDP-based network protocol enables real-time communication.

The machine learning setup uses semantically segmented lanes of the streets as an input. These colored lanes can be directly produced by the simulation. In the real-world a machine learning based segmentation method is used to achieve the segmented lanes.

Two methods are used to train a controller (actor): Imitation learning as a supervised learning method in which a Stanley controller serves as a teacher. Secondly, Twin Delayed Deep Deterministic Policy Gradient (TD3) is used to minimize the Cross-Track Error (CTE) of the miniature vehicle with respect to its lateral position in the street. Both methods are applied equally in simulation and in the real-world and are compared.

Preliminary results show high accuracy in lane following and intersection navigation in simulation and real-world, supported by precise real-time feedback from the tracking system. While full integration of the RL model is ongoing, the presented results show the platform's potential to further investigate the sim-to-real aspects in autonomous driving.

1 INTRODUCTION

The article presents the current progress on a project on miniature autonomy, where various aspects of applying artificial intelligence methods in real-world scenarios are investigated using small autonomous model vehicles (Tiedemann et al., 2019), (Pareigis et al., 2021), (Tiedemann et al., 2022).

A miniature landscape on a scale of 1:87 with a road was constructed and small vehicles have been developed to drive fully or semi-autonomously on the road.

There are many similar projects on autonomous driving in a small scale for research and educational purposes (e.g. (TU-Braunschweig, 2020), (Paull et al., 2017)). In many of these projects, the corre-

sponding platforms and environments have standardised and well-defined formats so that suitable experiments on autonomous driving can be carried out under controlled laboratory conditions.

In the project described here, a particularly complex environment is deliberately chosen for image-based autonomous driving. Figures 1 and 2 show an excerpt from the model landscape.

Two overhead cameras are used to generate current status images of the miniature world, which provide input for a digital twin. The miniature world is completely modelled digitally in two dimensions to allow experiments in simulations.

In a previous work by the authors also a 3D model of the miniature world has been created in Unity (see figure 3), and a steering controller has been trained inside the Unity simulation using reinforcement learn-


^a  <https://orcid.org/0000-0002-7238-0976>



Figure 1: Excerpt from the model landscape for testing autonomous vehicles. The realistic complexity of the model with different road colours, road markings, trees, buildings, etc. can be seen. The functionality of the Mapbuilder tool, which is used to generate a precise simulation of the real test environment, is also visualised.



Figure 2: The figure shows the model world with streets, a hill in the centre, houses and a rural background image. At the top two cameras can be seen, each connected to an RPi, and lamps.

ing methods which are provided in the ML-agents package of Unity.

The primary objectives of the experimental setup, which includes the miniature environment, autonomous and semi-autonomous vehicles, along with their corresponding digital models and simulations, are to facilitate sim-to-real transfer in reinforcement learning. The setup aims to enhance the understanding of artificial intelligence methods, while providing a platform to test image processing techniques and software architectures for autonomous driving in real-world-like scenarios.

Recent investigations in the aforementioned project relate to the so-called simulation-to-reality gap, or sim-to-real gap.

Autonomous agents that are trained in the simulation do not function without difficulty in real-world environments. Problems are not only caused by the

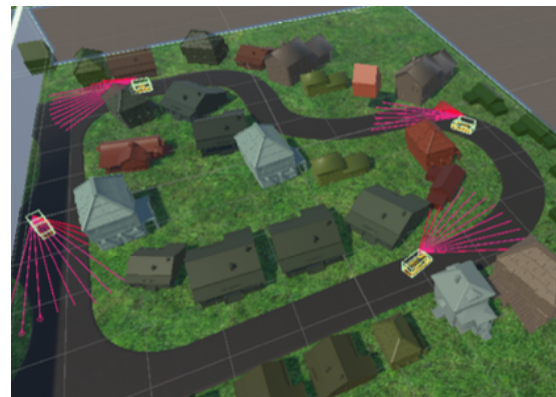


Figure 3: The illustration shows a model of the miniature world in Unity from a previous work. Vehicles are trained in Unity using ML-Agents and an array of distance scans. Training in simulation was successful. However, the results could not be applied to the real-world due to unavailability of distance information in the real miniature world.

different properties of sensor data in simulation and reality. The different physics and system dynamics in simulation and reality also generally cause difficulties.

In past studies (Pareigis and Maaß, 2023) a controller was successfully designed that compensates for changes in system dynamics during ongoing operation. The special architecture of the controller also makes it immune to discrepancies in system dynamics in simulation and reality. This controller was trained using imitation learning (IL) using a given PID controller as a template.

The reason for choosing a reinforcement learning (RL) based approach in this work is, that a teacher might not always be available. In addition, an IL-trained agent will never be able to perform superior to its teacher. Using RL, by specifying a target function to be optimized, given an adequate learning process, an improved driving behavior can ultimately be achieved.

The aim of the present investigations is to design a controller that optimizes a given target function and is robust against discrepancies in system dynamics in simulation and the real-world.

The paper describes the experimental setup, including

- the miniature model world
- the autonomous vehicle *TinyCar* and smart vehicles
- a digital twin composed of overhead cameras and tracking system
- the simulation system including map builder
- a machine learning architecture using simulation

and real-world data for supervised and reinforcement learning training

The paper describes the experimental setup of the miniature world including the vehicles and the communication infrastructure in section 2. The machine learning setup is described in section 3. Section 4 contains the experimental results with respect to real-time aspects, training and performance in simulation and real-world, supervised and reinforcement learning methods. Section 5 discusses the obtained results.

2 MINIATURE AUTONOMY SETUP

The section describes the setup of the model world, the tincar, the digital twin, the simulation environment and the reinforcement learning architecture.

2.1 Miniature Model World

Figure 2 shows the model world in scale 1:87. The model world features a variety of intersection types, curves of different diameter, various street surface colors, traffic lights, and various road edges. Milled wooden parts are used as a base for the road, in which magnets are inserted in a centre groove.

The embedded magnets enable automated miniature vehicles without active steering (which we call smart cars) to drive along the road. This allows traffic to be generated within which the autonomous miniature vehicles with active steering (Tincars) have to behave accordingly.

Switches are installed under the crossroads, which switch the grooves with the magnets and thus allow variable driving behaviour for the smart cars.

2.2 Tincar and Smart Cars

The tincar is a 1:87 scaled miniature vehicle designed to emulate the functionalities of a full-sized autonomous car. The primary components include a steering servo, a DC motor, LEDs for headlights and blinkers, and a front-facing camera. These components are controlled by an ESP32 microcontroller, chosen for its low power consumption and integrated Wi-Fi capabilities.

Figures 4 and 5 show the Tincar in the miniature environment and as a CAD drawing.

Automatic Repositioning. The reset procedure in the simulated environment repositions the car to a random spawn point. In the real-world, an automatic repositioning method is implemented to avoid manual



Figure 4: Tincar shown in Miniature Wonderland in Hamburg. The front facing camera is integrated into the windshield. A single LED is used as a headlight. The Tincar is also equipped with blinkers on each side.

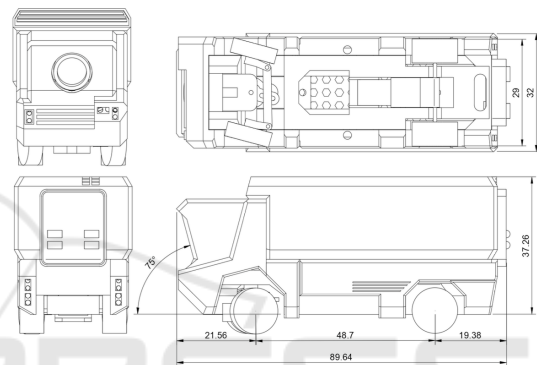


Figure 5: The CAD drawing of the Tincar shows its dimensions. The servomotor for controlling the steering is installed vertically to act directly on the steering linkage and to minimise play. The camera is installed at a slightly inclined angle.

intervention. This method continuously samples the nearest edge from the ground truth trajectory based on the car's current position and orientation. The car repositions itself by first driving backwards to clear obstacles, then forwards to the nearest edge using the Stanley control algorithm.

Smart Cars. Smart cars have been developed using the Faller Car System as a platform. These vehicles use a magnet attached to the steering linkage that keeps the vehicle on the magnetic track underneath the surface of the street. Other than the Faller Car System, the Smart Cars have been enhanced with a microprocessor that allows a WLAN-connection to control the vehicle's motor. Some Smart Cars are also equipped with a camera.

2.3 Digital Twin and Tracking System

The digital twin serves as a virtual replica of the miniature world, providing a controlled environment for the initial training of the reinforcement learning model.

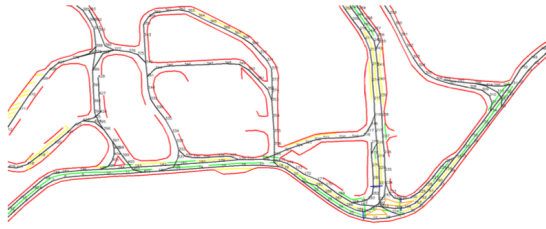


Figure 6: A map of part of Knuffingen in Miniatur Wunderland Hamburg is used for both simulation and real-world environments. It is manually annotated with a custom mapbuilder, creating a JSON file with lane line graphs and the ground truth trajectory. Node IDs on the map indicate the ground truth trajectory and define spawn points for training the reinforcement learning algorithm.

Overhead Cameras. Two high-resolution cameras are mounted above the track, offering a view of the entire driving environment. The cameras are attached to the top aluminium frame as can be seen in figure 2.

Tracking System. A tracking system is designed to provide continuous and precise feedback on the tiny-car’s position and orientation.

2.4 Simulation Environment

A simulation is used for initial training of the machine learning models. It serves for both training reinforcement learning and supervised learning algorithms.

The simulation consists of two main components: The car’s kinematics and the camera/lane segmentation simulation. These components operate on detailed maps, which are 1:1 replicas of the described laboratory environments and part of the miniature city (Knuffingen) in Miniatur Wunderland Hamburg (Miniatur Wunderland Hamburg, 2024).

Map Creation. The maps are manually annotated using a custom *mapbuilder*, which overlays lane lines and the ground truth lane path (center of the lane) onto a reference image. The mapbuilder generates a JSON file containing nodes and edges for each lane line and the ground truth trajectory, which is treated as a directional graph. This file is used by the simulation to render simulated camera images and calculate the car’s position and orientation after each step.

Figure 1 shows part of the map created by *mapbuilder* superimposed on an image of the overhead camera of the real miniature world.

Kinematics. The preliminary simplified objective of these investigations is to control only the steering angle, with a fixed speed at 0.5 m/s, allowing the simulation to focus on kinematics only.

The steering angle determines the curve radius, enabling the use of a linear transformation matrix to update the vehicle’s position and orientation.

Camera Rendering. In the simulation, the camera

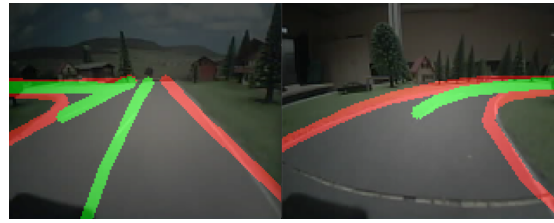


Figure 7: Two example images from the simulation are overlaid onto frames from the tiny-car’s real camera. Simulated images are generated for these positions and overlaid with the real camera feed to verify simulation accuracy.

rendering process generates lane segmentation images rather than raw camera frames. This approach simplifies the simulation and reduces computational requirements. The resulting images as produced by the simulation can be seen in figure 7.

Lane Segmentation. The lane segmentation model outputs binary images for each class of lane lines. These images are combined to form the observation space for the reinforcement learning agent.

Projection. The lane line points are projected into the camera frame using intrinsic and extrinsic camera matrices. This projection ensures that the simulated camera images closely match those captured by the real tiny-car.

3 REINFORCEMENT LEARNING ARCHITECTURE

This section outlines the reinforcement learning (RL) architecture used to train the tiny-car for autonomous navigation in both simulated and real-world environments. Key components include the RL algorithm, observation and action spaces, reward signal, and model architecture.

3.1 Twin Delayed Deep Deterministic Policy Gradient (TD3)

The TD3 algorithm (Fujimoto et al., 2018) controls the car’s steering angle. TD3, an off-policy actor-critic method, extends DDPG by using two critic networks to reduce overestimation bias and stabilize training. The actor network is optimized to maximize cumulative rewards, while the critics minimize the error between estimated and actual Q-values. Target policy smoothing and delayed policy updates further enhance training stability.

To promote exploration, temporally correlated Ornstein-Uhlenbeck noise is added to the actions, preventing premature convergence and encouraging diverse action selection (Lillicrap et al., 2016).

3.2 Observation and Action Spaces

3.2.1 Observation Space

The observation space is generated from preprocessed camera frames segmented into lane lines and road edges. A lane segmentation model outputs binary images for relevant road features, reducing the complexity of raw data processing and minimizing the sim-to-real gap. The observation space is defined as:

$$\mathcal{S} = \{0, 1\}^{C \times W \times H} \quad (1)$$

where C is the number of classes of segmented lane markings, W is the frame width, and H the height.

In addition to lane following, the agent receives a one-hot encoded maneuver vector m_t for specific actions such as turning. The policy is defined as:

$$\pi: \mathcal{S} \times \mathcal{M} \rightarrow \mathcal{A} \quad (2)$$

3.2.2 Action Space

The action space consists of the car's steering angle, with throttle held constant. The steering angle is a continuous value between -1 and 1, scaled by the maximum steering angle of the car:

$$a_t \in \mathcal{A} = [-1, 1] \quad (3)$$

3.2.3 Reward Signal

The reward is based on the cross-track error (CTE), encouraging the agent to minimize the deviation from the lane center:

$$R = \sum \gamma \cdot r_t \text{ with } r_t = \max(-33 \times \text{abs}(\text{CTE}) + 1, -1) \quad (4)$$

The value -33 is derived by dividing the maximum reward (1) by the maximum allowable cross-track error (CTE) of 0.03 meters, ensuring that any CTE beyond this threshold results in a negative reward.

An episode terminates if the car leaves the track, collides with an obstacle, or reaches the maximum of 1000 steps. A track exit occurs when the CTE exceeds 0.1 meters, and a collision is defined as the car remaining stationary for 10 steps despite having a set velocity. In the real-world environment, the car is automatically repositioned within the lane, while in simulation, it respawns at a random predefined point.

3.3 Model Architecture

The model is trained in two stages: pretraining the encoder using Imitation Learning (IL) and training the Actor and Critic networks using TD3.

Pretraining the Encoder with Imitation Learning (IL). The encoder is pretrained to extract features

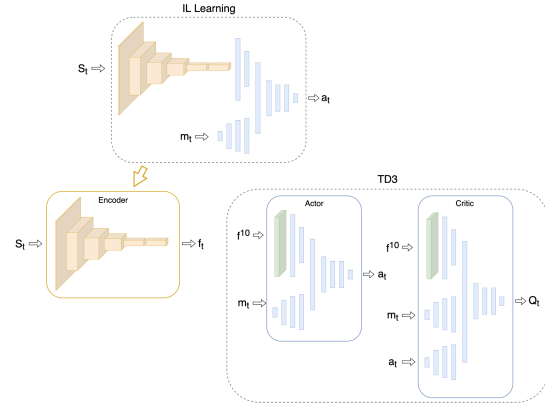


Figure 8: The Imitation Learning (IL) network pretrains the encoder, which is then used in TD3 training. The encoder outputs feature vectors that serve as input to the Actor and Critic networks, while the encoder remains frozen during TD3 updates.

from the observation space using data from a human driver or Stanley controller. This pretrained encoder is used for the Actor and Critic networks, without further updates during TD3 training, reducing computational load and speeding up learning.

Two IL models are tested:

- One trained entirely in simulation.
- Another finetuned with real-world data to reduce the sim-to-real gap.

Training the Actor and Critic Networks with TD3.

The Actor and Critic networks serve distinct roles but share a common feature extraction process. Both use a stack of the last 10 feature vectors as input, providing temporal context to smooth steering actions and prevent oscillations, similar to the derivative term in PID controllers.

The **actor** learns the policy by mapping observations to actions, maximizing the Q-value predicted by the critic. The **critic** estimates the Q-values of the actor's actions, with two critic networks in TD3 to reduce overestimation bias and stabilize training.

While both networks use the same pre-trained encoder, the actor optimizes action selection, while the critic evaluates those actions.

4 EXPERIMENTAL RESULTS

Extensive quantitative tests and experiments have been made with the described setup. This section gives a summary on the main results and achievements.

4.1 Real-Time Aspects

The training environment must meet real-time requirements for reinforcement learning, as neural network weights are updated during car interactions. The tinycar’s firmware uses FreeRTOS for strict timing, while the tracking and gym systems run on Linux and macOS.

Tinycar Latency. Frame latency is measured using a three-way handshake to account for clock synchronization issues. Network latency is defined as the time from frame capture to receipt in the training environment, adjusted for round-trip time (RTT). The total latency for a TD3 learning step is around 101 ms at 320x160 resolution, including compression, decompression, and inference.

Tracking System Latency. Latency is measured from frame capture by the overhead camera to its availability in the training environment, including OpenCV processing and network transmission. Tracking data (12 bytes) latency is calculated using $RTT/2$. Higher camera resolutions increase processing time.

4.2 Machine Learning Architecture

The section describes test results with respect to the machine learning architecture.

Encoder. The encoder compresses a 160×128 image into a 256-dimensional vector for the actor. It is trained via supervised learning using steering angles collected from manual driving or the Stanley controller. Three training setups were tested to minimize the sim-to-real gap:

1. Two encoders: one trained on simulation data, the other on real data.
2. One encoder trained on mixed simulation and real data.
3. Two encoders sharing actor weights: one trained on simulation data, and then used to train the second encoder on real data.

The simulation dataset contains 80,000 images, and the real dataset consists of 36,184 images, collected at HAW Hamburg and Knuffingen.

Feature Vector Comparison. A sequence of 631 frames was collected using the Stanley controller, with frames from the real camera (post-processed via lane segmentation) captured in parallel with simulated frames at the same positions. Feature vectors were computed for each encoder, and the mean, maximum, and standard deviation of the differences between simulated and real-world vectors were calculated (see Table 1). Encoders sharing actor weights

Table 1: Feature vector differences between real and simulated environments for different encoder setups. Lower values indicate higher similarity.

| Setup | Mean | Max | Std Dev |
|--------------|-------|--------|---------|
| Independent | 1.143 | 10.084 | 0.071 |
| Mixed | 0.679 | 7.486 | 0.093 |
| Shared Actor | 0.636 | 6.035 | 0.056 |

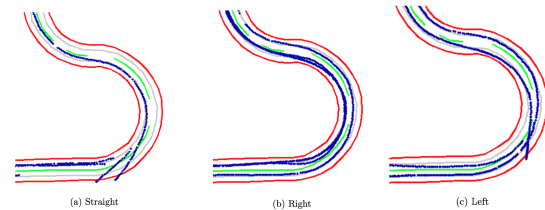


Figure 9: Real-world trajectory showing the right-turn maneuver maintaining lane control, while left and straight maneuvers struggle during right-hand turns.

produced the most similar feature vectors, with a mean difference of 0.636, compared to independent training.

Driving Performance. The encoder and actor were benchmarked in the real-world using 5000 steps per maneuver (straight, left, right). Table 2 shows that the right-turn maneuver had the lowest average CTE, similar to the Stanley controller. Real-world performance was slightly better than in simulation, demonstrating the encoder’s ability to predict steering angles accurately based on real input.

Table 2: Driving performance for the shared actor setup in real-world and simulation environments. CTE values in millimeters.

| Env | Maneuver | CTE Avg | CTE Std Dev |
|------|----------|---------|-------------|
| Real | Straight | 13.572 | 19.033 |
| | Right | 9.622 | 8.613 |
| | Left | 14.289 | 18.601 |
| Sim | Straight | 13.512 | 14.314 |
| | Right | 11.690 | 6.879 |
| | Left | 17.297 | 19.477 |

Trajectory Analysis. Figure 9 shows the driving paths for right and left maneuvers in the real-world. The right-turn maneuver handles both turns without leaving the lane, while the left and straight maneuvers struggle with right-hand turns, sometimes leaving the lane. The simulation did not exhibit this behavior, but wide turns at intersections led to higher CTE in both environments.

4.3 Reinforcement Learning Training Results

As described in Section 3.1, the actor is trained using the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. The training begins in simulation to establish baseline behavior, followed by real-world fine-tuning to adapt to physical dynamics and address the sim-to-real gap. Experiments have shown that using encoders with shared weights from the supervised learning actor leads to optimal performance, and this configuration is employed for training the TD3 actor.

The replay buffer has a capacity of 500,000, and the critic is trained after every step using a batch size of 256. The actor is updated every second step with the same batch size. Learning rates are set to $2e-4$ for the critic and $1e-4$ for the actor, with a target update rate $\tau = 0.001$ and a discount factor $\gamma = 0.99$. Each episode has a maximum of 1,000 steps, with randomized maneuvers and spawn points.

A linear reward shaping strategy is applied, where rewards range from 1 to -1 based on the cross-track error (CTE). A CTE of 60 mm results in the minimum reward, and the episode terminates if the CTE exceeds 70 mm for 5 consecutive steps. In the real-world environment, an additional termination occurs if the car's velocity drops below 0.01 m/s for 5 steps, preventing the vehicle from becoming stuck.

Initial testing of the actor occurs in simulation to verify autonomous driving capability and fine-tune hyperparameters. Subsequently, real-world testing is conducted on the track at HAW Hamburg. Due to the absence of tracking data in the Knuffingen environment, this setup is only used in simulation. After initial testing with simulation-trained weights, the actor continues real-world training, and both versions are compared. Additionally, the encoder is retrained using the TD3 actor's weights to assess potential performance gains by leveraging simulation-specific features.

RL in Simulation. The actor undergoes 1,000 training episodes in the simulation, with a maximum of 1,000 steps per episode in the HAW environment and 2,000 steps in the larger Knuffingen environment. The reward curves show improvement over time, though not reaching maximum rewards, indicating some residual CTE.

The right-turn maneuver consistently exhibits the lowest CTE and heading error (H-Error). In the Knuffingen environment, the actor demonstrates oscillations and misalignment at intersections, resulting in early termination when the CTE exceeds 70 mm.

RL in Real-World. Due to the absence of a tracking system in the Knuffingen environment, real-world

testing is limited to the HAW Hamburg environment. The TD3 actor trained in simulation is first tested in the real world without modification, except for replacing the encoder with one trained on real-world data.

After initial tests, the actor is trained for an additional 300 episodes in the real world, using the same hyperparameters as in simulation. The simulation training experience buffer is retained to avoid catastrophic forgetting. Results indicate that real-world performance is generally worse than in simulation, with higher CTE and H-Error across all maneuvers. Training in the real world improved left-turn performance but degraded right-turn performance, with no significant change in straight maneuvers.

Subsequent tests revealed that using raw camera data instead of lane segmentation did not enhance performance, underscoring the importance of accurate real-world observation space for the TD3 actor.

4.4 Discussion on Real-World RL

This section analyzes the experimental results and overall project. The primary objective was to train a reinforcement learning (RL) model in a simulated environment and successfully transfer it to the real world. The model was expected to autonomously navigate the car, execute maneuvers at intersections, and adapt to real-world conditions.

4.4.1 Environment

The training environment integrated both a car simulation and a digital twin for real-world operations. Real-time performance and the tracking system were critical to the RL task. While the simulated camera closely replicated the real camera, differences in lane segmentation were observed, likely due to limited training data. The tracking system's latency and accuracy were generally sufficient, but enhancements are needed to reduce false positives.

4.4.2 Encoder

The encoder reduced the high-dimensional image input into a 256-dimensional feature vector. Several training setups were evaluated to reduce the sim-to-real gap, with the shared actor weights technique producing the most similar feature vectors. However, direct minimization of feature vector differences could offer further improvements. Although the encoder trained with real-world data enabled the simulation-trained actor to drive the car, difficulties were noted in left and straight maneuvers.

4.4.3 Actor

The TD3 actor, initially trained in simulation and then in the real world, performed better in the simpler HAW environment compared to Knuffingen. Real-world training, supported by the simulation experience buffer, prevented catastrophic forgetting but still underperformed relative to simulation. While the supervised-trained actor achieved reasonable real-world results, the TD3 actor struggled, likely due to the sim-to-real gap or limitations in lane segmentation input. Testing with raw camera data did not yield performance improvements, indicating a need for more robust models or alternative learning algorithms for real-world applications.

5 CONCLUSION

This paper presented the development of a comprehensive system for testing and training machine learning (ML) and reinforcement learning (RL) models for autonomous driving within a miniature environment. The system features a digital twin of a 1:87 scaled city model at HAW Hamburg, enabling simulated tests and real-world training for RL methods.

The system employs overhead cameras for tracking the position and orientation of a 1:87 scale autonomous vehicle, the tiny-car, in real time. The tiny-car, equipped with a front-facing camera for environmental perception, demonstrates effective control via a low-latency wireless camera stream, supporting real-world RL experiments.

Initial results show that the system is capable of training neural networks to autonomously navigate and handle intersections. The high-precision tracking system, combined with automatic repositioning, achieves an 84% episode reset success rate without human intervention. The encoder successfully reduces the input data dimensionality, minimizing the sim-to-real gap and shortening the training time required in real-world environments. Models trained with supervised learning demonstrate effective performance, particularly in handling intersections.

However, RL models trained in simulation exhibit challenges when transferred to more complex real-world environments. While intersection handling in simulation is reliable, real-world performance, particularly in the Knuffingen environment, reveals issues such as oscillations and difficulties in lane selection at intersections. The real-world transfer of models shows limited generalization and marginal improvement after additional training, indicating the need for further refinement.

Overall, this work provides a functional framework for investigating RL in real-world settings and addressing the sim-to-real gap. Future research will focus on improving the tracking system, extending the gym environment to encompass more complex scenarios, and exploring advanced perception methods, such as autoencoders, to further reduce the sim-to-real gap and enhance the real-world applicability of simulation-trained models.

REFERENCES

- Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1587–1596. PMLR.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*.
- Miniatur Wunderland Hamburg (2024). Knuffingen: Miniatur wunderland. <https://www.miniatur-wunderland.de/wunderland-entdecken/welten/knuffingen/>. Accessed: 2024-09-29.
- Pareigis, S. and Maaß, F. L. (2023). Improved robust neural network for sim2real gap in system dynamics for end-to-end autonomous driving. In Gini, G., Nijmeijer, H., Burgard, W., and Filev, D., editors, *Informatics in Control, Automation and Robotics*, volume 836 of *Lecture Notes in Networks and Systems*, pages 1–21. Springer, Cham.
- Pareigis, S., Tiedemann, T., Kasten, M., Stehr, M., Schnirpel, T., Schwalb, L., and Burau, H. (2021). Künstliche Intelligenz in der Miniaturautonomie. In *Fachtagung des gemeinsamen Fachausschusses Echtzeitsysteme von Gesellschaft für Informatik e.V. (GI), VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik (GMA) und Informationstechnischer Gesellschaft im VDE (ITG) 2020*, Informatik aktuell, pages 41–50.
- Paull, L., Tani, J., Knepper, R., Rus, D., and Leonard, J. (2017). Duckietown: an open, inexpensive and flexible platform for autonomy education and research. *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1497–1504.
- Tiedemann, T., Fuhrmann, J., Paulsen, S., Schnirpel, T., Schönherr, N., Buth, B., and Pareigis, S. (2019). Miniature autonomy as one important testing means in the development of machine learning methods for autonomous driving : How ml-based autonomous driving could be realized on a 1:87 scale. In *International Conference on Informatics in Control, Automation and Robotics 2019, ICINCO 2019 : proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics*, pages 483–488.
- Tiedemann, T., Schwalb, L., Kasten, M., Grotkasten, R.,

and Pareigis, S. (2022). Miniature autonomy as means to find new approaches in reliable autonomous driving ai method design. *Frontiers in neurorobotics*, 16:846355.

TU-Braunschweig (2020). Carolo-cup: An autonomous driving competition. In *Proceedings of the 2020 Autonomous Driving Competitions*.

