

# A Platform-Agnostic Deep Reinforcement Learning Framework for Effective Sim2Real Transfer in Autonomous Driving

Dianzhao Li<sup>1,2\*</sup> and Ostap Okhrin<sup>1,2</sup>

<sup>1</sup>Chair of Econometrics and Statistics, esp. in the Transport Sector, Technische Universität Dresden, Dresden, 01187, Germany.

<sup>2</sup>Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany.

\*Corresponding author(s). E-mail(s): [dianzhao.li@tu-dresden.de](mailto:dianzhao.li@tu-dresden.de);  
Contributing authors: [ostap.okhrin@tu-dresden.de](mailto:ostap.okhrin@tu-dresden.de);

## Abstract

Deep Reinforcement Learning (DRL) has shown remarkable success in solving complex tasks across various research fields. However, transferring DRL agents to the real world is still challenging due to the significant discrepancies between simulation and reality. To address this issue, we propose a robust DRL framework that leverages platform-dependent perception modules to extract task-relevant information and train a lane-following and overtaking agent in simulation. This framework facilitates the seamless transfer of the DRL agent to new simulated environments and the real world with minimal effort. We evaluate the performance of the agent in various driving scenarios in both simulation and the real world, and compare it to human players and the PID baseline in simulation. Our proposed framework significantly reduces the gaps between different platforms and the Sim2Real gap, enabling the trained agent to achieve similar performance in both simulation and the real world, driving the vehicle effectively.

**Keywords:** DRL, Sim2Real, Lane following, Overtaking

# 1 Introduction

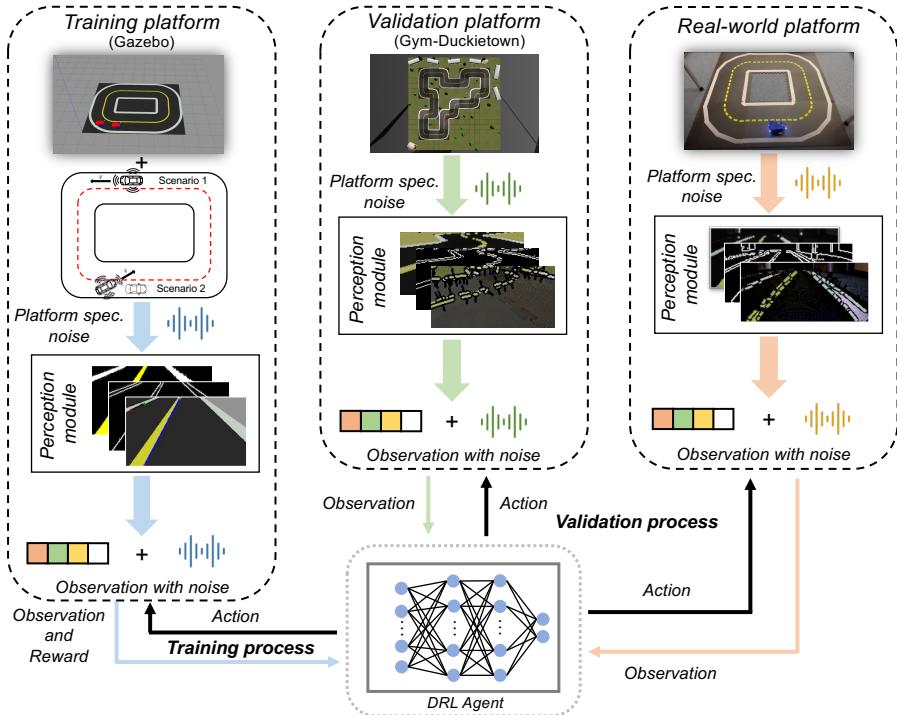
Recently, there has been an encouraging evolution in Deep Reinforcement Learning (DRL) for solving complex tasks [1–11]. Consequently, DRL is increasingly attractive for use in transportation scenarios [12, 13], where vehicles must make continuous decisions in a dynamic environment, and behaviors in the real-world traffic can be extremely challenging.

Fully autonomous driving consists of multiple simple driving tasks, for instance, car following, lane following, overtaking, collision avoidance, traffic sign recognition, etc. Simple tasks and their combination need to be tackled to build cornerstones in achieving fully autonomous driving [14–16]. For this purpose, researchers use DRL algorithms to solve various tasks, such as lane keeping [17], lane changing [18], overtaking [19], and collision avoidance [20]. Additionally, to improve driving performance, researchers often combine human prior knowledge with DRL algorithms [21, 22].

Developing a fully autonomous driving system with DRL algorithms requires training an agent in a simulator and implementing it in real-world scenarios. Since training a DRL agent directly in the real world is impractical due to the enormous number of interactions required to improve policies, simulation plays here a crucial role. Despite the increasing realism of simulators, there are significant differences between simulation and reality that create a Sim2Real gap, making the transfer of DRL policies from simulation to the real world challenging [23]. To address this issue, several techniques in different research fields such as domain randomization, domain adaptation, knowledge distillation, meta reinforcement learning, and robust RL have been proposed [11, 24–30] in particular to narrow the gap between the simulation and reality. In transportation research, Sim2Real methods already enable the agents to effectively perform tasks such as lane following [31–33] and collision avoidance maneuvers [34, 35] in the real world.

The aforementioned works are mainly focused on a single driving task such as lane following. To the best of our knowledge, there are no related works for overtaking, not to mention the combination of both, the lane following and overtaking tasks and transformation to the real world. This research gap motivated us to propose a robust DRL agent training framework for Sim2Real with the capabilities of following the lane and safely overtaking slower vehicles.

The proposed framework shown in Fig. 1 comprises a platform-dependent *perception module* and a universal DRL *control module*. The *perception module* acts as an abstraction layer that addresses the heterogeneity between different platforms and enhances generalization by extracting task-relevant affordances from the traffic state. The affordances relevant to the lane following and overtaking tasks are then produced as the input state for the *control module*. The latter is the DRL agent that utilizes the aggregated information from the *perception module* and trains its driving policy within the simulator for different driving scenarios. To this end, we use a Long Short-Term Memory (LSTM) based DRL algorithm that has been proven to be effective for multiple driving tasks, such as lane following and overtaking [36–40]. The DRL



**Fig. 1** Robust Sim2Real transfer with DRL agent and platform-dependent perception module that separates the agent from the environment. The DRL agent is first trained in the Gazebo simulator with the information provided by the perception module in Gazebo, afterwards, the trained agent is evaluated in Gym-Duckietown environment, and the real-world scenario with real Duckiebots without additional effort but only with platform specified parameterized perception module. During the training process, two different driving scenarios for the agent to tackle, are lane following (Scenario 1) and overtaking (Scenario 2) tasks.

agent is first trained on a simple circular map within the Gazebo simulator [41], then directly transferred for the validation to another environment namely the Gym-Duckietown environment [42] and finally tested in the real-world scenario without any additional efforts. Due to the separation of the perception module and the control module, the extracted affordances are utilized by the DRL algorithm, therefore, the generalization of the trained agent is ensured [43]. Summarizing, the main contributions of our work are as follows:

- The training framework with the separation of the *perception module* and *DRL control module* is proposed to narrow the reality gap and deal with the visual redundancy. It enables the agent to be transferred into real-world driving scenarios and across multiple simulation environments seamlessly.
- Due to the exceptional generalization property of the proposed framework, the DRL is trained in a *simple map in one simulator* and transferred to more complex maps within other simulators and even real-world scenarios.

- The developed autonomous vehicle, which employs the proposed DRL algorithm, demonstrates the ability to successfully perform lane following and overtaking maneuvers in both simulated environments and real-world scenarios, showcasing the robustness and versatility of the approach.
- The agent shows superior performance compared to the baselines within the simulations and real-world scenarios. Human baselines are collected within the Gym-Duckietown environment for the lane following task, highlighting the super-human performance of the DRL agent.

## 2 Results

In this section, we compare the performance of the trained DRL agent with benchmarks in both evaluation and real-world environments and show the results with multiple metrics.

### 2.1 Performance metrics

*Lane following evaluation in simulation:* For quantitatively demonstration of the performance, five metrics are used to compare the lane following capabilities in simulation: (1) survival time ( $T_s$ ), the simulation is terminated when the vehicle drives outside of the road, (2) traveled distance ( $d_t$ ) along the right lane of evaluation episodes, (3) lateral deviation ( $\delta_l$ ) from the right lane center, (4) orientation deviation ( $\delta_\phi$ ) from the tangent of the right lane center, and (5) major infractions ( $i_m$ ), which are measured in the time spent outside of the drivable zones. Both deviations  $\delta_l$  and  $\delta_\phi$  are integrated over time.

We consider the objective of lane following task to be driving the vehicle within the boundaries of the right lane with minimal lateral and orientation deviations. In order to provide an exhaustive evaluation and to make a more informative comparison between agents, a final score is used to evaluate the lane following ability. This score takes into account the performance on each of the individual metrics, weighting them appropriately to provide a comprehensive and fair evaluation of the overall performance of the agents:

$$Score = T_s + d_t - \frac{1}{2}\delta_l - \frac{3}{2}\delta_\phi - \frac{3}{2}i_m. \quad (1)$$

The traveled distance and survival time, are set as the primary performance metrics for the lane following task [44]. Furthermore, the penalty is given when the vehicle is driven with deviation from the right lane center or even completely on the left lane. The penalty weights are determined based on the severity of the undesired behaviors. While driving, it is important to avoid the left lane as much as possible. Additionally, the lateral deviation should be kept within a small tolerance. Orientation deviation is less critical but still should be minimized. Worth mentioning is, that function (1) is not used as the reward function for the DRL agent, but purely for comparison and evaluation purposes.

*Overtaking evaluation in simulation:* In addition to evaluating lane following capability, we also assess the overtaking ability of the trained agent. Thus,

we add one more performance metric, *success rate of overtaking maneuvers* to the validation process, which is calculated as the ratio between the number of successful overtaking actions and the total number of overtaking maneuvers initiated by the agent. We also retain the other performance metrics to indicate whether the agent can perform lane following under the influence of overtaking behavior.

*Evaluation in real-world environment:* For the lane following evaluation in the real-world scenario, only four metrics are selected to evaluate the performance: lateral deviation ( $\delta_l$ ) and orientation deviation ( $\delta_\phi$ ) measured from the perception module, average velocity, and infractions. While the lateral and orientation deviations are not exact measurements, they provide a rough impression of staying within the lane and maintaining the correct orientation. The infraction metric is triggered when the agent veers off the road and requires human intervention to return to the correct path.

## 2.2 Evaluation maps

During the training phase, we utilize a simple circular map depicted in training platform of Fig. 1. However, for evaluation purposes, we employ more challenging maps to test the robustness of the trained agent. Specifically, for the Gym-Duckietown environment, we use five different maps, namely Normal 1, Normal 2, Plus track, Zig-Zag, and V track, as illustrated in Fig. 2a, to evaluate the lane following capabilities of the agents [42]. These five maps allow us to test the ability of the agents to maintain lanes in various driving scenarios, including straight lines, curves, and continuous curves. To evaluate the overtaking performance of the agents in simulation, we use only three tracks, namely Normal 1, Normal 2, and Zig-Zag, which are selected due to their relatively straight segments, making them suitable for overtaking maneuvers. Finally, to evaluate the performance in real-world scenarios, we used the circular map depicted in real-world platform of Fig. 1 for both lane following and overtaking tasks.

## 2.3 Lane following in simulation

This section presents an evaluation of lane following ability for the trained DRL agent in the Gym-Duckietown environment. We compare the DRL agent with two benchmarks: the classical proportional–integral–derivative (PID) controller [45] and the human driving baseline. In the validation, the PID baseline controller obtains precise information such as the relative position and orientation of the vehicle to the right lane center directly from the simulator. This information is not available to the DRL agent or human players, as discussed in Section 4.3. The human baseline is driven by human players to perform lane following maneuvers on five evaluation maps, see Section 4.4. The DRL agent utilizes processed information from the perception module to generate control commands for the robot, as detailed in Section 4.6. The PID controller [45], however, has access to the simulator, therefore it can be regarded as a

near-optimal controller for the lane following task in the simulation. To assess the robustness of the lane following capability of DRL agent, we introduce a feature that scales its action output, allowing for switching between *fast* and *slow driving modes* with different driving preferences. This feature enables us to distinguish between a sporty and calm driver.

**Table 1** Evaluation results for different approaches within different tracks during the lane following evaluation in simulation, best performance for each track are highlighted.

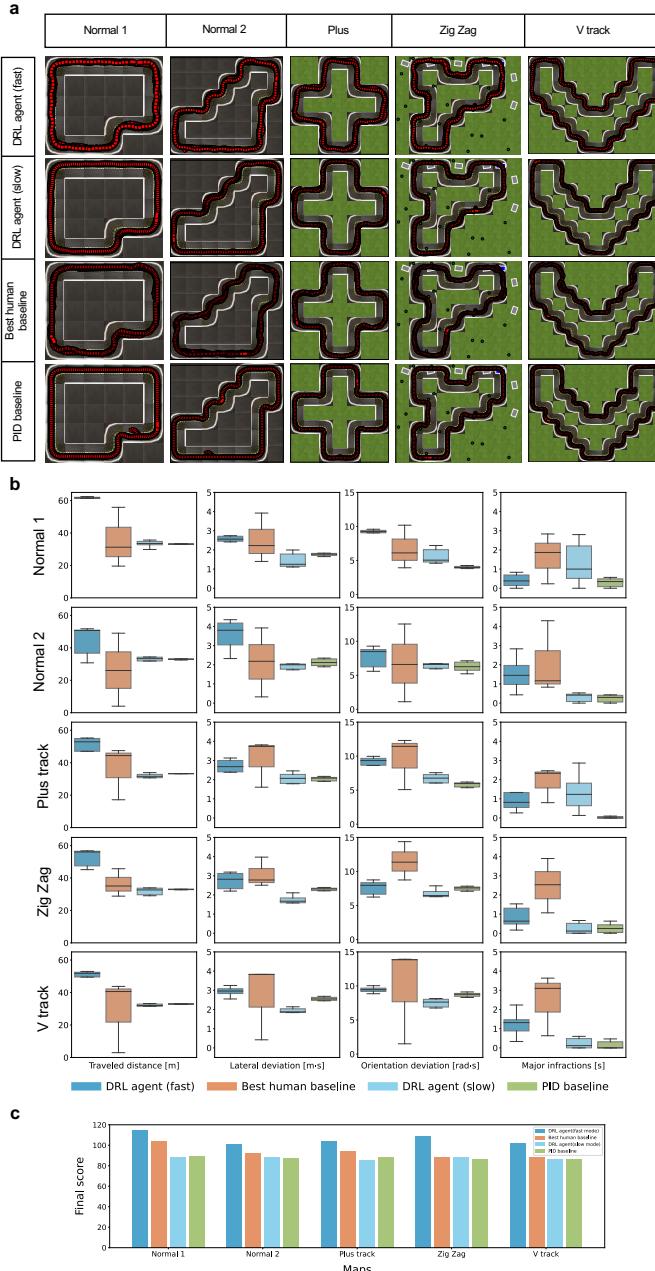
Maps	Median metric over 100 episodes	Agents			
		DRL slow	agent <sup>1</sup> fast	PID <sup>2</sup> baseline	Human <sup>3</sup> baseline
Normal 1	Final score	88.50	<b>114.52</b>	89.18	104.01
	Survival Time ( $T_s$ ) [s] <sup>4</sup>	60	60	60	60
	Traveled distance ( $d_t$ ) [m]	33.75	<b>62.27</b>	33.17	55.82
	Lateral deviation ( $\delta_l$ ) [m·s]	<b>1.25</b>	2.56	1.69	3.92
	Orientation deviation ( $\delta_\phi$ ) [rad·s]	5.04	9.24	<b>3.86</b>	10.18
	Major infractions ( $i_m$ ) [s]	0.99	0.38	<b>0.25</b>	1.87
Normal 2	Final score	88.15	<b>105.40</b>	87.64	92.39
	Survival Time ( $T_s$ ) [s]	60	60	60	60
	Traveled distance ( $d_t$ ) [m]	34.10	<b>53.42</b>	32.99	49.03
	Lateral deviation ( $\delta_l$ ) [m·s]	2.99	2.54	<b>2.02</b>	3.92
	Orientation deviation ( $\delta_\phi$ ) [rad·s]	6.60	8.87	<b>6.07</b>	12.54
	Major infractions ( $i_m$ ) [s]	<b>0.17</b>	0.70	0.20	4.30
Plus track	Final score	85.21	<b>104.30</b>	87.91	93.86
	Survival Time ( $T_s$ ) [s]	60	60	60	60
	Traveled distance ( $d_t$ ) [m]	32.51	<b>52.88</b>	33.14	47.45
	Lateral deviation ( $\delta_l$ ) [m·s]	<b>2.07</b>	2.68	2.10	3.74
	Orientation deviation ( $\delta_\phi$ ) [rad·s]	6.77	9.35	<b>5.96</b>	12.31
	Major infractions ( $i_m$ ) [s]	1.23	0.82	<b>0.10</b>	2.47
Zig-Zag	Final score	87.88	<b>108.72</b>	87.77	88.70
	Survival Time ( $T_s$ ) [s]	60	60	60	60
	Traveled distance ( $d_t$ ) [m]	32.95	<b>56.46</b>	33.98	45.71
	Lateral deviation ( $\delta_l$ ) [m·s]	<b>1.67</b>	2.82	2.24	3.97
	Orientation deviation ( $\delta_\phi$ ) [rad·s]	<b>6.45</b>	7.96	7.14	14.38
	Major infractions ( $i_m$ ) [s]	<b>0.12</b>	0.63	0.27	3.90
V track	Final score	86.52	<b>102.40</b>	85.74	88.32
	Survival Time ( $T_s$ ) [s]	60	60	60	60
	Traveled distance ( $d_t$ ) [m]	32.38	<b>51.77</b>	32.94	43.76
	Lateral deviation ( $\delta_l$ ) [m·s]	<b>1.87</b>	2.95	2.63	3.82
	Orientation deviation ( $\delta_\phi$ ) [rad·s]	<b>7.63</b>	10.15	9.15	13.92
	Major infractions ( $i_m$ ) [s]	0.12	0.90	<b>0.0</b>	3.10

<sup>1</sup>By adjusting the scale of action output, we can switch between fast and slow driving modes.

<sup>2</sup>The PID baseline uses the exact information from the simulator, which is not available for the other agents.

<sup>3</sup>For the human baseline, we took the best performance.

<sup>4</sup>The maximum evaluation time in one episode is 60s.



**Fig. 2** Evaluation results in Gym-Duckietown. **a**, Illustration of sampled vehicle trajectories for different approaches within five different maps. **b**, Boxplots of different performance metrics for different approaches during the evaluation process. **c**, Final scores comparison between the DRL agent and other baselines, where the fast-mode DRL agent achieves the highest score in every evaluation map and is followed by the best human player, slow-mode DRL agent, and PID baseline.

The DRL agent and PID baseline are evaluated in 100 episodes across five different maps (Section 2.2), with the vehicle being spawned at random positions and orientations before each episode to assess the robustness of their lane following capability. The median value of their performance across the episodes is used as the evaluation statistic. For the human baseline, however, only the best performance is used for comparison. The evaluation results for the DRL agent, best human baseline, and PID controller are shown in Table 1 and Fig. 2. The exemplary vehicle trajectories from evaluation episodes for each agents are depicted in Fig. 2a. The vehicle trajectories of the slow-mode DRL agent are as smooth as the near-optimal PID baseline controller. For a more quantitatively analyze, Table 1, Fig. 2b, and Fig. 2c show that the fast-mode DRL agent achieves the best performance in every evaluation map regarding the final score and traveled distance. Compared with the near-optimal PID baseline based on the traveled distance, namely the velocity, the DRL agent drives almost 50~70% faster. The DRL agent also outperforms the best human baseline with faster speed and fewer deviations and infractions. Additionally, the slow-mode DRL agent performs even better than the near-optimal PID baseline controller with the same level of speed in terms of lateral and orientation deviation.

## 2.4 Overtaking in simulation

Within this section, we evaluate the overtaking capability of the trained agent in simulation, as there are no existing benchmarks for overtaking, we only access the DRL agent with the proposed metrics discussed in Section 2.1. Here only the slow-mode DRL agent is utilized for the overtaking task. Similar as for lane following, during the evaluation episodes, the ego vehicle and slower vehicle are spawned at random positions and orientations within three evaluation maps, and the success rate of all episodes are computed. In addition, we also monitor the lane following performance metrics to assess the lane following capabilities of agent under the influence of overtaking behaviors. The results are presented in Table 2.

**Table 2** Evaluation results of DRL agent for overtaking behavior on different maps.

Median metric over 10 episodes	Maps		
	Normal 1	Normal 2	Zig-Zag
Success rate <sup>1</sup>	94.74 %	94.44 %	90.91 %
Survival Time ( $T_s$ ) [s]	60	60	60
Traveled distance ( $d_t$ ) [m]	21.90	21.42	22.29
Lateral deviation ( $\delta_l$ ) [m·s]	1.59	2.16	2.79
Orientation deviation ( $\delta_\phi$ ) [rad·s]	5.33	7.30	8.88
Major infractions ( $i_m$ ) [s]	5.97	5.07	2.38

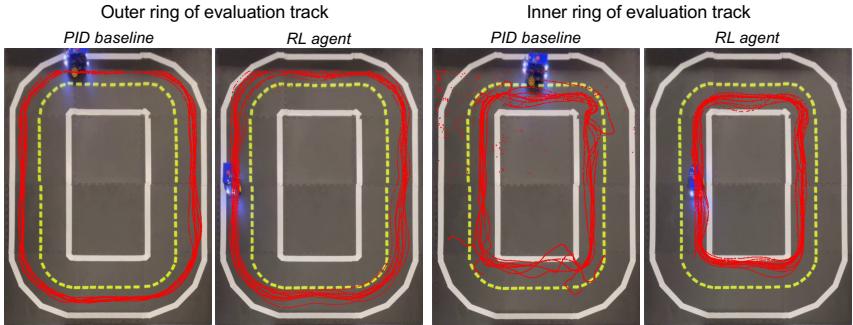
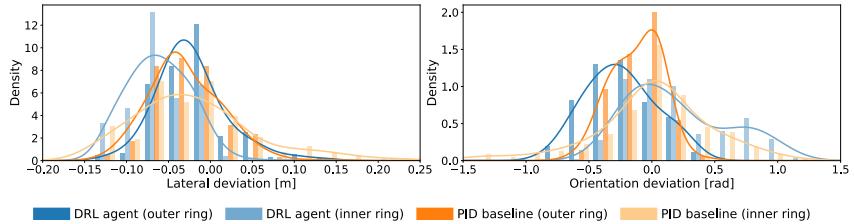
<sup>1</sup>The success rate is computed based on all 10 episodes.

Based on the evaluation results, we observe that the success rates of overtaking maneuvers for all three maps exceeded 90%. The lateral and orientation deviation levels remain consistent with those observed during the lane following evaluation, which suggest that the overtaking behaviors of agent did not compromise its lane following capabilities, and that it is able to maintain an adequate level of control during overtaking maneuvers. However, due to the need for the agent to overtake obstacles and drive on the left lane, the magnitude of infractions observed during the evaluation is greater than that seen during the pure lane following evaluation. Overall, the DRL agent can perform safe overtaking maneuvers when there are obstacles in front and within the detection range. After the overtaking behavior, the agent drives back to the right lane and resumes performing lane following.

## 2.5 Lane following in real-world environment

For the Sim2Real transformation, the real-world scenario is used to assess the lane following and overtaking abilities of the trained agent in the real world. For real-world lane following capabilities, we compare the DRL agent with the PID baseline discussed in Section 4.3. The PID baseline algorithm in this case utilizes the output information from the perception module to control the vehicle. For the DRL agent, the information from the perception module is used as part of the input state. To ensure fairness in the evaluation process, we conducted experiments with three different vehicles in the real-world scenario on both the inner and outer ring of the track for a duration of 180 seconds (Fig 3a). During the experiments, we logged the lateral deviation and orientation deviation from the perception module, as well as the velocity of the vehicle. Additionally, we captured videos of the evaluation process and performed subsequent Image Processing to analyze the performance of the two approaches.

The real world trajectories of the vehicles and the distribution of the lateral and orientation deviation results for different approaches during the evaluation are depicted in Fig. 3. In Table A1, the metrics of the two approaches are compared, with the best results being highlighted. For the PID baseline, the average velocity of the vehicle is the maximal possible velocity the baseline algorithm can drive without keeping driving off the road. The evaluation results presented in Table A1 show that, for each test vehicle, the DRL agent outperforms the PID baseline with respect to the average speed, which is the most important performance metric. Specifically, the DRL agent achieved up to a 65% faster average speed than the PID baseline controller. Moreover, the DRL agent had only one infraction during the evaluation for all three vehicles. Furthermore, as depicted in Fig. 3, the DRL agent achieved the same or better levels of lateral and orientation deviations compared to the PID controller.

**a****b**

**Fig. 3** Evaluation results for lane following in real-world scenario for one vehicle, the other two are shown in Fig. A1. **a**, Illustration of the vehicle trajectories for different approaches i.e. DRL agent and PID baseline on the inner and outer ring of the real-world track. **b**, Distribution histogram and kernel density estimate (KDE) plots of the lateral and orientation deviation for PID baseline and DRL agent during the real-world lane following evaluation.

## 2.6 Overtaking in real-world environment

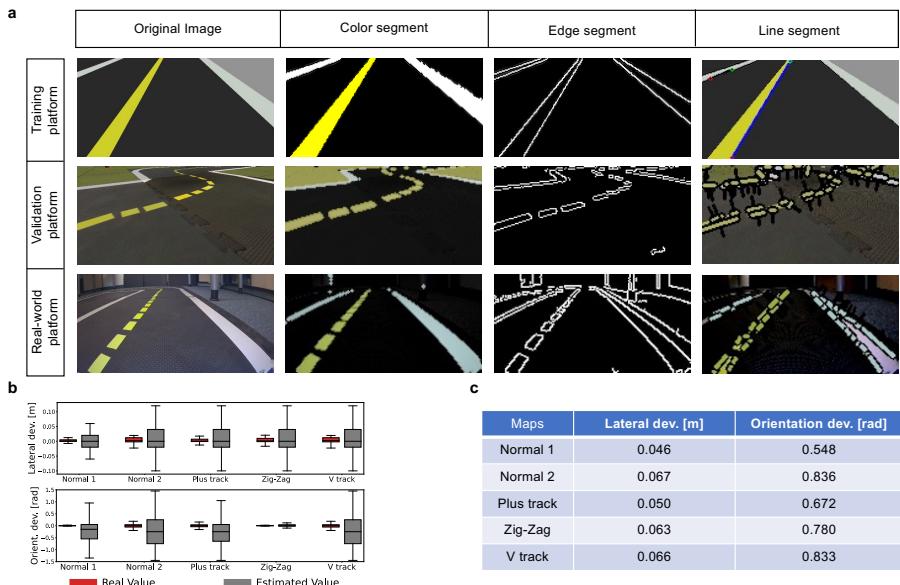
Same as in the simulation, the overtaking capability of the trained DRL agent in the real world is also assessed. The vehicle used during the lane following is upgraded with LIDAR equipment for a more wide range of distance detection. For the LIDAR equipment, we utilize an RPLIDAR A1M8, which is a low-cost 2D laser scanner solution that can perform 360° scanning within 12m range. We use the same track in the lane following evaluation, see Fig 3a. As there is no baseline controller for overtaking maneuvers of the real vehicle, we only collect the results of the trained DRL agent.

In the evaluation of overtaking performance, a slower vehicle is guided by a PID baseline for lane following, while a faster vehicle is under the control of a DRL agent. Once the slower vehicle is detected within the detection range, the *overtaking mode* is initiated and the DRL agent executes the overtaking maneuver. Once the overtaking is completed, the agent promptly maneuvers the vehicle back to the right lane and resumes lane following. In addition to the successful overtaking behavior, the DRL agent is also equipped to handle unsatisfactory overtaking situations, where it may drive off the road after passing the slower vehicle. In such cases, the agent is capable of recovering and guiding the vehicle back on track to resume lane following, demonstrating

the robustness of the DRL agent. A video showcasing both successful and unsatisfactory overtaking behaviors is available on the project website.

## 2.7 Perception module

To conduct a comprehensive evaluation of the performance of the DRL agent, it is crucial to consider not only the control module but also the perception module used by the agent. The output image obtained during the multi-step image processing procedure detailed explained in Section 4.2 is shown in Fig. 4a. However, in the real-world scenario, the true values of the lateral and orientation deviation from the perception module are not available. Thus, we only evaluate the perception module in the Gym-Duckietown environment, as shown in the boxplots in Fig. 4b. Additionally, to assess the performance of the perception module across different maps, we calculate the root mean square error (RMSE), as presented in Fig. 4c. The RMSE ranges from 0.046m to 0.067m, which is relatively large given that the width of the road is only 0.23m.



**Fig. 4** Output results from perception module for different platforms. **a**, Different outputs during the multi-steps perception module, i.e. the image input from the camera, the detected road markings according to the color with HSV thresholding, the detected marking edges with canny filter, and final detection marking used to estimate the lateral displacement  $d$  and angle offset  $\phi$ . **b**, The comparison between the real value and the estimated value from the perception module in the validation simulator concerning the lateral and orientation deviation. **c**, The prediction root mean square error (RMSE) of the perception module regarding the lateral and orientation deviation over different maps.

Despite the inaccurate and variable output information of the perception module for the DRL agent, it exhibits superior performance compared to other benchmarks. This suggests that the DRL agent is robust and can handle sub-optimal input states from the perception module.

### 3 Discussion

Sim2real is a critical step towards fully autonomous driving using DRL, but discrepancies between different frameworks are hindering this progress. Thus, we propose a training framework that facilitates the transformation of the trained agent between two different platforms, e.g., Sim2Real transfer. The proposed framework includes the perception module with platform-dependent parameters and the control module with the DRL algorithm. The platform-dependent parametric perception module improves the generalization of the overall system and simplify the transformation of the trained DRL agent. It also enables the agent to adapt to different environments and improves its ability to handle variations in input data. For the control module, we employ LSTM-based DRL algorithms, which can effectively handle time-dependent decision-making tasks such as lane following and overtaking behaviors.

To validate our framework, we conducted multiple evaluations in both simulated and real-world environments. First, we validate our trained DRL agent in a Gym-Duckietown environment to perform lane following and overtaking behaviors. Despite the challenges posed by the poor performance of the perception module, the DRL agent still outperforms benchmarks such as PID and trained human players, demonstrating the effectiveness and robustness of our framework. We also evaluate the DRL agent in a real-world environment (Sim2Real), where we only modified the parameters in the perception module and applied the trained control module directly. Compared to the PID baseline controller, the trained agent achieved higher speed and fewer infractions during the evaluation, demonstrating its practical applicability. Additionally, we successfully show the overtaking capability of one faster vehicle in the real-world environment.

While our framework has shown promising results in bridging the gap between different platforms and facilitating Sim2Real transfer, there are still some limitations that call for further improvements. One such limitation is that the framework currently treats vehicles as black boxes and does not account for their dynamic differences. The control module outputs high-level commands, such as velocity, which may not be suitable for other robotic systems. Additionally, for the overtaking task, the vehicle in front is driven with a constant slow velocity, which is not a realistic scenario. A more realistic approach would be to implement a cooperative overtaking behavior where the leader recognizes the overtaking intention of the follower and decides to slow down or maintain a constant velocity to facilitate the overtaking action. This can be achieved using multi-agent reinforcement learning (MARL), which will be the focus of our future research.

## 4 Methods

### 4.1 Problem formulation

Achieving fully autonomous driving requires addressing fundamental driving tasks such as car following, lane following, and overtaking. In this work, we focus specifically on lane following and overtaking maneuvers on a two-lane road (see Fig. 1). Our autonomous agent is designed to maintain lane keeping behavior using information collected by sensors when no vehicle is driving in front or no static obstacle is on the road. When a slower-moving vehicle or a static obstacle blocks the path, the agent must safely overtake the obstacle. To achieve successful transfer of learned behavior from the simulated environment to the evaluation simulator and the real-world environment, the agent must be capable of accounting for any discrepancies between platforms. Moreover, the trained agent must be able to drive the vehicle in the real-world environment without colliding with other vehicles.

To enable the agent to perform these tasks, we equipped the ego vehicle with multiple sensors, including a camera for lane detection and distance sensors for vehicle detection, as described in Section 4.7.

### 4.2 Perception module

In this section, we will describe in detail the multi-step image processing pipeline employed in the perception module, as illustrated in Fig. 4a. The pipeline starts with compensating for the variability in illumination. Initially, the k-means clustering algorithm is used to detect the primary clusters on the road from a subsample of sensor pixels. These clusters are then matched with the expected red, yellow, white, and gray clusters based on prior information, which may slightly vary depending on the environment. An affine transformation is then fitted in the RGB color space between the detected clusters and the color-balanced version of the image to obtain the illumination-corrected image. The line segmentation detection pipeline is then executed using a Canny filter to detect edges [46], and Hue-Saturation-Value (HSV) colorspace thresholding is employed to detect the expected colors in parallel. The resulting output images are displayed in the second and third columns of Fig. 4a. Next, individual line segments are extracted using a probabilistic Hough transform [47], and the resulting output images are depicted in the fourth column of Fig. 4a. The length and orientation of line segments are used to filter out noise and extract the relevant lines. Reprojection from image space to the world frame is performed based on extrinsic and intrinsic calibration [48]. Using different line segments, a nonlinear non-parametric histogram filter [49] is used to estimate the lateral displacement  $d$  to the right lane center and the angle offset  $\phi$  relative to the center axis. These estimates are utilized later in the control module to facilitate lane following and overtaking behaviors [44].

### 4.3 PID controller

The PID controller is a classical control theory that has been widely used in the domain of automatic control [45]. Due to its simplicity and effectiveness, it has become a well-established method for controlling a variety of systems, including those in the field of robotics and autonomous vehicles. Its widespread use in industry and academia has led to a deep understanding of its strengths and limitations, making it a valuable benchmark for comparison with newer control methods, such as DRL.

In this study, the proportional-derivative (PD) controller is used as a benchmark for comparison with the performance of the DRL agent in simulation. The PD controller uses information about the relative location and orientation of the robot to the centerline of the right lane, which is available in the simulator. We conduct a series of tests to evaluate the performance of the PD controller using the same information from the perception module as the DRL agent. However, the results showed poor performance of the PD controller, which can be attributed to the uncertainty in the output from the perception module. To achieve more stable control, the velocity of the robot is kept constant, and the orientation  $\phi_t$  is controlled using PD control:

$$\phi_t = K_p \theta_{\delta,t} + K_d (\theta_{\delta,t} - \theta_{\delta,t-1}),$$

where  $\phi_t$  is the orientation output at time step  $t$  and  $\theta_{\delta,t}$  is the orientation deviation provided by the simulator. The proportional gain  $K_p$  and derivative gain  $K_d$  are set as 2.0 and 5.0, respectively.

In real-world lane following scenarios, a PI controller is used. However, precise information on lateral and orientation deviations is unavailable, and instead, features extracted from the perception module are used. The orientation control output in the real-world at time step  $t$ , denoted by  $\phi_{t,r}$ , is calculated using the following equation:

$$\phi_{t,r} = K_{p,r}^d d_{\delta,t,r} + K_{i,r}^d \sum_{i=0}^t d_{\delta,r}(i) + K_{p,r}^\theta \theta_{\delta,t,r},$$

where  $d_{\delta,t,r}$  and  $\theta_{\delta,t,r}$  indicate the lateral and orientation deviations from the perception module, respectively. The proportional coefficients  $K_{p,r}^d$  and integral coefficients  $K_{i,r}^d$  for lateral deviation are set as -6.0 and -5.0, respectively. The proportional coefficients for orientation deviation are denoted by  $K_{p,r}^\theta$  and set as -0.3. The parameters in the PID controller for both simulation and real-world are chosen based on multiple experiments [44].

### 4.4 Human baseline

To compare the lane following performance of the DRL agent with that of human players, we design a keyboard controller to control the robot in the Gym-Duckietown environment and collect data on the performance of human players. For perception, human players observe the same image output from the camera in front of the robot and use the human internal visual perception

function to locate its position. We recruit 25 human players, including 16 males and 9 females, with an average age of 31, who are mainly researchers in transportation or computer science, with varying levels of expertise in robotics and autonomous driving.

To ensure fairness and accuracy in the comparison between the DRL agent and human players, each human player is given a minimum of 20 minutes to train using five different maps that are later used for evaluation. This allows the human players to become familiar with the Gym-Duckietown environment and the perception module used in the study. During the evaluation, each human player has six attempts to drive within the evaluation tracks, and only the best performance in terms of the final score (1) is recorded as their final result. This is done to ensure that the human players are given a fair chance to perform at their best, without the pressure of having only one attempt.

For transparency and further analysis, the overall data of human players, including their scores and trajectories, are collected and available on the project webpage. In addition, a leaderboard is created to display the top-performing human players, allowing for easy comparison with the performance of the DRL agent.

## 4.5 Vector field guidance (VFG) method

The Vector Field Guidance (VFG) method is a widely used approach for solving the path following problem [50]. In this work, we adapt the VFG method to facilitate the lane following task of the DRL agent. Specifically, we calculate the desired course angle command based on the current position of the ego vehicle and the desired lane configuration as

$$\chi^d = \chi^\infty \cdot \arctan(ke_y) + \chi^{path},$$

where  $\chi^\infty$  and  $k$  represent the maximum course angle variation for path following and convergence rate tuning parameters respectively. And  $\chi^{path}$  indicates the direction of the target path defined in the inertial reference frame. The cross track error  $e_y$  of the vehicle to the target path can be computed as follows

$$e_y = \sin \left\{ \chi^{path} - \arctan \left( \frac{y_{av} - y_{k-1}}{x_{av} - x_{k-1}} \right) \right\} \cdot d_{w_{k-1}},$$

with  $d_{w_{k-1}} = \sqrt{(y_a - y_{k-1})^2 + (x_a - x_{k-1})^2}$ , where  $(x_a, y_a)$  and  $(x_{k-1}, y_{k-1})$  are the coordinates of the agent and the closest point on the target path to the position of the agent respectively.

## 4.6 Control module

This section introduces the setup used to train various DRL agents for the control module, and subsequently compares their performance to obtain an optimal driving policy.

#### 4.6.1 Observation and action space

As previously mentioned, our DRL agent uses the lateral displacement  $d_\delta$  and angle offset  $\theta_\delta$  as observation states. Additionally, we employ the VFG method as guidance states to assist the agent in following the optimal trajectory. However, for the overtaking task, the agent needs to be aware of other vehicles in its vicinity. To address this, we include the time-to-collision with other vehicles as a conditional observation state. When other vehicles are detected within the range, this state is activated and works as a regular input state. Otherwise, it is set to zero. Furthermore, we perform input state normalization to speed up the learning process and enable faster convergence.

The overall input state  $s_t$  for the DRL agent at time step  $t$  is thus defined as

$$s_t = \left( \frac{d_{\delta,t}}{d_w}, \frac{\theta_{\delta,t}}{\pi}, v_t, \frac{\chi_{yaw,t}}{2\pi}, e_{y,t}, \rho_l T_c, a_{v,t-1}, a_{\phi,t-1} \right)^\top, \quad (2)$$

where  $d_w$  denotes the lane width,  $v_t$  denotes the velocity of the ego vehicle, and  $\chi_{yaw,t}$  defines the angle offset between the current heading and the desired heading in the optimal trajectory,  $e_{y,t}$  indicates the cross-track error. Here,  $\rho_l$  is a binary flag indicating whether there is another vehicle within the detection range  $c_d$ . Additionally, we use the time-to-collision  $T_c = d_{v,t}/v_t$  between the ego vehicle and another vehicle, where  $d_{v,t}$  represents the distance between the two vehicles. The action space of our DRL agent consists of the speed command  $v_{c,t}$  and steering angle  $\phi_t$ . Note that the VFG method input states  $\chi_{yaw,t}$  and  $e_{y,t}$  are only used during the training process. For evaluation, the VFG is not available, we replace  $\chi_{yaw,t}$  and  $e_{y,t}$  with  $\theta_\delta$  and  $d_\delta$  from perception module. The hyperparameters used for the input states and reward function are shown in Table 3.

**Table 3** Hyperparameters for input state and reward function.

Parameter	Value
Lane width ( $d_w$ )	0.23 m
Detection range( $c_d$ )	0.4 m
Reward factor ( $a$ )	0.001
Sensitivity parameter $k_{r_c}$	0.6

#### 4.6.2 Reward function

The reward function formulation is a crucial aspect of the RL algorithm as it defines the desired behaviors of the agent. In our case, the objective of the agent is to follow the right lane and overtake slower vehicles when necessary. Therefore, we design the reward function to balance the driving efficiency, lane following, and overtaking capability of the agent:

$$R_t = \begin{cases} -1, & \text{if collide or out of boundary,} \\ w_c R_{c,t} + w_v R_{v,t} + w_\theta R_{\theta,t}, & \text{otherwise.} \end{cases} \quad (3)$$

The agent obtains a penalty of -1 when collisions have occurred or if drove off the road. Otherwise, weighted positive rewards are given. The sub-reward functions are shown in Table 4. The first term,  $R_{c,t}$ , guides the agent to learn lane following ability by utilizing the cross-track error  $e_{y,t}$  towards the target path, where  $k_{r_c}$  is a tunable parameter that controls sensitivity. The second term  $R_{v,t}$  devises for driving efficiency, and for overtaking behavior, the agent should leverage the positive velocity reward and collision penalty and obtain suitable overtaking behavior. The third term,  $R_{\theta,t}$ , penalizes the agent for driving in the opposite direction. As the weights measure the trade-off of different sub-rewards, we conduct many experiments, test different combinations of weights, and select the set of weights that has the best performance. The weights for lane following part  $w_c = 0.3$ , efficiency or overtaking factor  $w_v = 0.6$  and the heading error weights  $w_\theta = 0.1$ . It is important to note that this reward function is only used during the training process and is not the same as the score (1) used during the evaluation.

**Table 4** Sub-rewards function used for DRL agents.

Reward Term	Functions
Lane following reward	$R_{c,t} = a^{k_{r_c} e_{y,t}}$
Velocity bonus	$R_{v,t} = \frac{v_t - v_{desired}}{v_{desired}}$
Heading error reward	$R_{\theta,t} = \theta_t - \theta^*$

## 4.7 Simulated and real environment

In this work, Robot Operating System (ROS) and Gazebo simulator are used as our simulation platform to train the agent. ROS is a widely used open-source robotics middleware suite that offers a set of software frameworks for robot development [51]. It enables the communication between various sensors on a single robot and multiple robots using a node-to-node communication mechanism. Gazebo is an open-source 3D robotics simulator with real-world physics in high fidelity. It integrates the ODE physics engine, OpenGL rendering, and supports code for sensor simulation and actuator control. Combining both results into a powerful robot simulator with real vehicle dynamics enables simpler Sim2Real transfer. To assess the robustness of the proposed agent across multiple simulated platforms, Gym-Duckietown is utilized as the validation simulator [42]. We used ROS wrappers in the validation simulator to

minimize the efforts for the transformation. For validation in real-world scenarios, we use Duckiebot as the base and upgrade with LIDAR equipment for a more wide range of distance detection. The Duckiebot is equipped with a front-facing 160deg FOV camera, hall effect sensor wheel encoders, Inertial Measurement Unit (IMU), and time of flight (tof) sensor. For the LIDAR equipment, we utilize an RPLIDAR A1M8. The computation unit used in the robot is NVIDIA Jetson Nano 2GB.

## 4.8 Training process

During the training process, we introduce a surrounding vehicle with low velocity that performs lane following using the PID baseline algorithm. The optimal trajectories for VFG guidance are collected using this vehicle. To further enhance the performance of the ego vehicle in handling both static and dynamic obstacles, the surrounding vehicle will occasionally stop for five seconds and then continue driving. The ego vehicle, which is driven by the DRL agent, is trained to perform overtaking behavior when it is close to the slower surrounding vehicle. When there is no surrounding vehicle in front, the ego vehicle is trained to maintain driving within the right lane. This setup helps to ensure that the ego vehicle learns to handle different traffic scenarios during training.

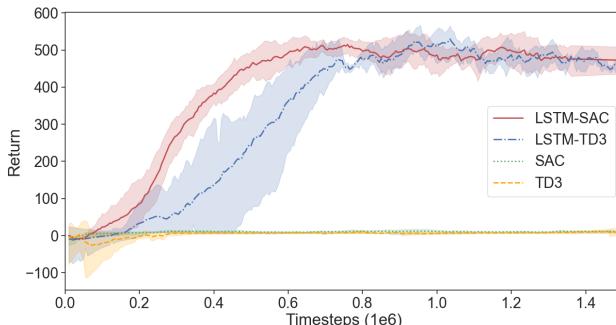
In this work, TD3, SAC, LSTM-TD3, and LSTM-SAC with the hyperparameters in Table 5 are trained [52–55]. Vanilla TD3 and SAC are trained for comparison with LSTM-based TD3 and SAC. The training processes are carried out with an NVIDIA GeForce RTX 3080 GPU and take roughly forty hours to finish one million timesteps of interaction with the simulated environment. The DRL agents are trained for 1.5 million timesteps with 10 different initial seeds and their training performance is evaluated by computing the average test return of 10 evaluation episodes, which is exponentially smoothed. Fig. 5 shows the training performance of all agents, where we observe that the two LSTM-based agents converge successfully during the training process. Although the LSTM-SAC agent converges faster and exhibits greater stability than the LSTM-TD3 agent, they achieve similar performance at the end of the training. However, the TD3 and SAC agents fail to learn and obtain approximately zero rewards, even though they use the same reward function as the LSTM-based agents. This result suggests that the recurrent neural networks are essential for the overtaking task in this work. Since only the distance to the surrounding vehicle is used for the overtaking task, the agent needs the past information to avoid confusion with the same distance observation during the overtaking phases.

**Table 5** List of hyperparameters used for DRL agents in the training process.

Hyperparameter	Value
Neural network structure <sup>1</sup>	$2 \times [256, \text{ReLU}]$
Loss function	MSELoss
Training batch size ( $N$ )	100
Replay buffer size ( $\mathcal{D}$ )	$10^5$
Actor learning rate ( $l_{r,a}$ )	$10^{-4}$
Critic learning rate ( $l_{r,c}$ )	$10^{-4}$
Reward discount factor ( $\gamma$ )	0.99
Target update rate ( $\tau$ )	0.005
Update start step	5000
Optimizer	Adam
Policy update delay <sup>2</sup>	2
History length for LSTM agent	2

<sup>1</sup>The activation functions of network outputs for Actor and Critic are *Tanh* and *Linear* respectively.

<sup>2</sup>Policy update delay is used in TD3 and LSTM-TD3.



**Fig. 5** Test return obtained by the different DRL agents during the training process.

#### 4.9 Fairness versus baseline approaches

Comparing the DRL agent to other baseline approaches may not be entirely fair, given that humans and algorithms approach tasks differently, with different perceptions and decision-making processes. To ensure a fair evaluation, we consider various aspects.

First is the perception method used by different agents. During the evaluation in the Gym-Duckietown environment, we compare the PID baseline and human players with the DRL agent. The PID baseline uses exact information about its position within the lane, which is not available to the other approaches, giving it a significant advantage. However, when validating in real-world scenarios where the PID baseline and DRL agent have the same level of observation, the performance of the PID baseline is inferior to that of the DRL

agent. Human players rely on simple observation to determine their position and react accordingly, which is arguably a distinct advantage to DRL agent.

Next, we turn our attention to vehicle control. Both the PID baseline and DRL agent employ the same action space, while human players use a keyboard to manipulate the vehicle’s speed, allowing them to constantly adjust its velocity. As a result, there is no clear advantage among the three approaches.

Moving on to driving strategy, we first note that the PID baseline aims to minimize deviations in current velocity, which may be a minor disadvantage when it comes to maximizing the final score. On the other hand, human players are provided with information on how to achieve a higher score through the final score equation, which may incentivize them to take shortcuts and drive at high speeds, a practice we do not encourage. Finally, while the reward function of the DRL agent requires it to strike a balance between deviations and velocity, the weights in the final score equation differ from those in the reward function. Therefore, we do not believe that the DRL agent has an inherent advantage when it comes to driving strategy.

## Declarations

### Data availability

The data including statistical results and trajectories collected from human players, baseline agents, and DRL agent have been made available at <https://dailyl.github.io/sim2realVehicle.github.io/>. The recorded videos of evaluation results in the simulation and real world are also available.

### Code availability

All code to train the DRL agent was implemented in Python using deep learning framework Pytorch. The code supports the plot within this paper are also available at [https://github.com/DailyL/Sim2Real\\_autonomous\\_vehicle](https://github.com/DailyL/Sim2Real_autonomous_vehicle). We have provided all the source code under the MIT license, making it open and accessible to anyone interested in building upon our work.

## Appendix A Extended Data

## References

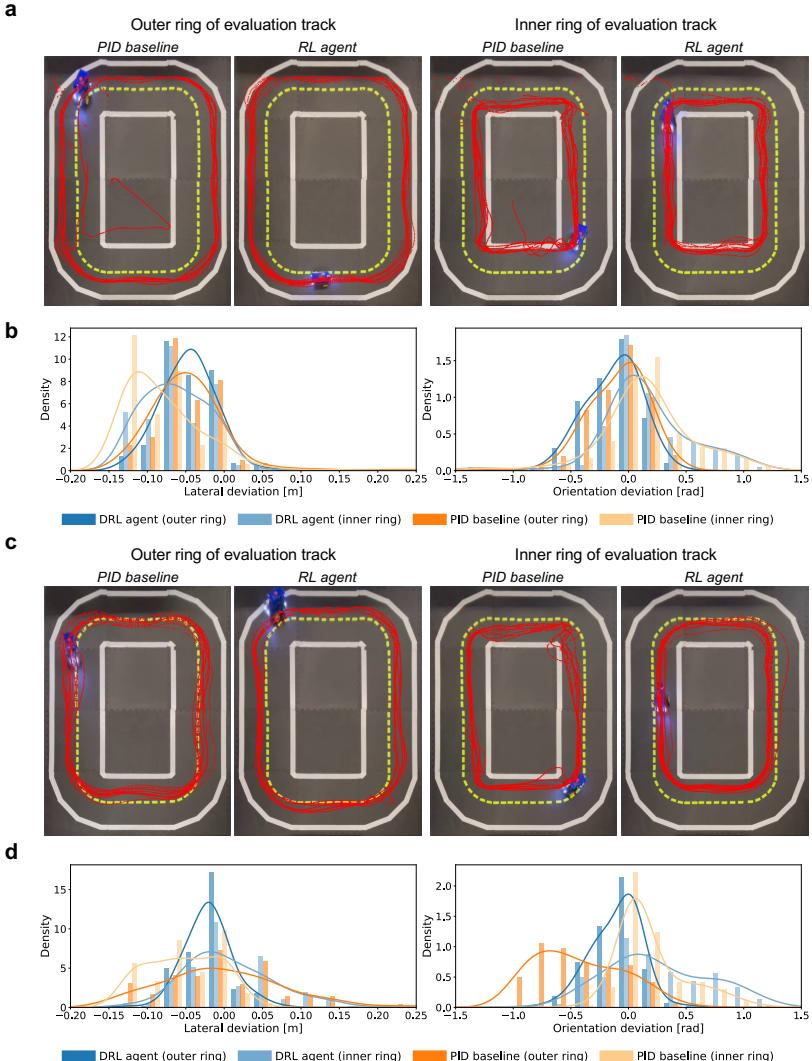
- [1] Vinyals, O. *et al.* Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* **575** (7782), 350–354 (2019) .
- [2] Bellemare, M. G. *et al.* Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature* **588** (7836), 77–82 (2020) .
- [3] Wurman, P. R. *et al.* Outracing champion gran turismo drivers with deep reinforcement learning. *Nature* **602** (7896), 223–228 (2022) .

**Table A1** Evaluation results for real-world lane following task with different vehicles, best results of every performance metric for each vehicles are highlighted.

Vehicles	Driving direction	Control algorithm	Lateral deviation ( $\delta_l$ ) <sup>1</sup>		Orientation deviation ( $\delta_\phi$ ) <sup>1</sup>		Average Velocity [m/s]	Infraction <sup>2</sup>
			Mean [m]	Stdv. [m]	Mean [rad]	Stdv. [rad]		
Vehicle 1	Outer ring	PID baseline	-0.0301	0.0427	<b>-0.1208</b>	<b>0.2064</b>	0.4369	0
		DRL agent	<b>-0.0232</b>	<b>0.0423</b>	-0.2496	0.2808	<b>0.6999</b>	0
	Inner ring	PID baseline	<b>-0.0194</b>	0.0716	<b>0.0107</b>	0.4513	0.4371	8
		DRL agent	-0.0609	<b>0.0375</b>	0.1993	<b>0.4059</b>	<b>0.6004</b>	0
Vehicle 2	Outer ring	PID baseline	-0.0467	0.0509	<b>-0.0944</b>	0.2776	0.4399	2
		DRL agent	<b>-0.0466</b>	<b>0.0359</b>	-0.1292	<b>0.2401</b>	<b>0.7289</b>	0
	Inner ring	PID baseline	-0.0790	0.0512	0.1654	0.4050	0.4400	3
		DRL agent	<b>-0.0614</b>	<b>0.0446</b>	<b>0.1597</b>	<b>0.4039</b>	<b>0.6056</b>	1
Vehicle 3	Outer ring	PID baseline	<b>-0.0024</b>	0.0773	-0.4321	0.3821	0.4413	0
		DRL agent	-0.0209	<b>0.0312</b>	<b>-0.1033</b>	<b>0.2160</b>	<b>0.6423</b>	0
	Inner ring	PID baseline	-0.0528	0.0471	<b>0.1898</b>	<b>0.3031</b>	0.4427	2
		DRL agent	<b>0.0079</b>	<b>0.0618</b>	0.2461	0.4507	<b>0.5814</b>	0

<sup>1</sup>The lateral and orientation deviation in this table are not the exact value in the real world but the output from the perception module.

<sup>2</sup>Infraction in the real-world evaluation is counted whenever the agent drives off the road and needs to be relocated on the track by a human operator.



**Fig. A1** Evaluation results of other two vehicles for lane following in real-world scenario. **a** and **c** are the illustration of the vehicle trajectories for different approaches i.e. DRL agent and PID baseline on the inner and outer ring of the real-world track for two vehicles respectively. **b** and **d** are the distribution histogram and kernel density estimate (KDE) plots of the lateral and orientation deviation for PID baseline and DRL agent during the real-world lane following evaluation for two vehicle respectively.

- [4] Agostinelli, F., McAleer, S., Shmakov, A. & Baldi, P. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence* **1** (8), 356–363 (2019).
- [5] Fawzi, A. *et al.* Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **610** (7930), 47–53 (2022).

- [6] Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518** (7540), 529–533 (2015) .
- [7] Degraeve, J. *et al.* Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **602** (7897), 414–419 (2022) .
- [8] Shin, D. *et al.* Deep reinforcement learning-designed radiofrequency waveform in mri. *Nature Machine Intelligence* **3** (11), 985–994 (2021) .
- [9] Xue, C. *et al.* Phy-q as a measure for physical reasoning intelligence. *Nature Machine Intelligence* **5** (1), 83–93 (2023) .
- [10] Won, D.-O., Müller, K.-R. & Lee, S.-W. An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions. *Science Robotics* **5** (46), eabb9764 (2020) .
- [11] Andrychowicz, O. M. *et al.* Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* **39** (1), 3–20 (2020) .
- [12] Talpaert, V. *et al.* Exploring applications of deep reinforcement learning for real-world autonomous driving systems (2019). Preprint at <https://arxiv.org/abs/1901.01536>.
- [13] Kiran, B. R. *et al.* Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* **23** (6), 4909–4926 (2021) .
- [14] Geisslinger, M., Poszler, F. & Lienkamp, M. An ethical trajectory planning algorithm for autonomous vehicles. *Nature Machine Intelligence* 1–8 (2023) .
- [15] Orr, I., Cohen, M. & Zalevsky, Z. High-resolution radar road segmentation using weakly supervised learning. *Nature Machine Intelligence* **3** (3), 239–246 (2021) .
- [16] Pek, C., Manzinger, S., Koschi, M. & Althoff, M. Using online verification to prevent autonomous vehicles from causing accidents. *Nature Machine Intelligence* **2** (9), 518–528 (2020) .
- [17] Sallab, A. E., Abdou, M., Perot, E. & Yogamani, S. End-to-end deep reinforcement learning for lane keeping assist (2016). Preprint at <https://arxiv.org/abs/1612.04340>.
- [18] Wang, P., Chan, C.-Y. & de La Fortelle, A. A reinforcement learning based approach for automated lane change maneuvers. In IEEE Intelligent Vehicles Symposium (IV), pp. 1379–1384 (IEEE, 2018).

- [19] Kaushik, M., Prasad, V., Krishna, K. M. & Ravindran, B. Overtaking maneuvers in simulated highway driving using deep reinforcement learning. In IEEE intelligent vehicles symposium (IV) 1885-1890. (IEEE, 2018).
- [20] Ngai, D. C. K. & Yung, N. H. C. A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers. *IEEE Transactions on Intelligent Transportation Systems* **12** (2), 509–522 (2011) .
- [21] Vecerik, M. *et al.* Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards (2017). Preprint at <https://arxiv.org/abs/1707.08817>.
- [22] Li, D. & Okhrin, O. Modified DDPG car-following model with a real-world human driving experience with carla simulator. *Transportation Research Part C: Emerging Technologies* **147**, 103987 (2023) .
- [23] Zhao, W., Queralta, J. P. & Westerlund, T. Sim-to-real transfer in deep reinforcement learning for robotics: a survey (2020). In IEEE symposium series on computational intelligence (SSCI) 737-744(IEEE, 2020).
- [24] Zhu, Y. *et al.* Target-driven visual navigation in indoor scenes using deep reinforcement learning. In IEEE international conference on robotics and automation (ICRA) 3357-3364 (IEEE, 2017).
- [25] Tobin, J. *et al.* Domain randomization for transferring deep neural networks from simulation to the real world (2017). In IEEE/RSJ international conference on intelligent robots and systems (IROS) 23-30 (IEEE, 2017).
- [26] Traoré, R. *et al.* Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer (2019). Preprint at <https://arxiv.org/abs/1906.04452>.
- [27] Rusu, A. A. *et al.* Policy distillation (2015). Preprint at <https://arxiv.org/abs/1511.06295>.
- [28] Wang, J. X. *et al.* Learning to reinforcement learn (2016). Preprint at <https://arxiv.org/abs/1611.05763>.
- [29] Morimoto, J. & Doya, K. Robust reinforcement learning. *Neural computation* **17** (2), 335–359 (2005) .
- [30] Chebotar, Y. *et al.* Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In International Conference on Robotics and Automation (ICRA) 8973-8979 (IEEE, 2019).

- [31] Almási, P., Moni, R. & Gyires-Tóth, B. Robust reinforcement learning-based autonomous driving agent for simulation and real world (2020). In International Joint Conference on Neural Networks (IJCNN) 1-8 (IEEE, 2020).
- [32] Sazanovich, M., Chaika, K., Krinkin, K. & Shpilman, A. Imitation learning approach for ai driving olympics trained on real-world and simulation data simultaneously (2020). Preprint at <https://arxiv.org/abs/2007.03514>.
- [33] Sandha, S. S., Garcia, L., Balaji, B., Anwar, F. & Srivastava, M. Sim2real transfer for deep reinforcement learning with stochastic state transition delays (2021). In Conference on Robot Learning 1066-1083 (PMLR, 2021).
- [34] Morad, S. D., Mecca, R., Poudel, R. P., Liwicki, S. & Cipolla, R. Embodied visual navigation with automatic curriculum learning in real environments. *IEEE Robotics and Automation Letters* **6** (2), 683–690 (2021) .
- [35] Byravan, A. *et al.* Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. *arXiv preprint arXiv:2210.04932* (2022) .
- [36] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9** (8), 1735–1780 (1997) .
- [37] Altché, F. & de La Fortelle, A. An lstm network for highway trajectory prediction (2017). In IEEE 20th international conference on intelligent transportation systems (ITSC) 353-359 (IEEE, 2017).
- [38] Perot, E., Jaritz, M., Toromanoff, M. & De Charette, R. End-to-end driving in a realistic racing game with deep reinforcement learning (2017). In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops 3-4 (IEEE, 2017).
- [39] Su, S., Muelling, K., Dolan, J., Palanisamy, P. & Mudalige, P. Learning vehicle surrounding-aware lane-changing behavior from observed trajectories (2018). In IEEE Intelligent Vehicles Symposium (IV) 1412-1417 (IEEE, 2018).
- [40] Zhang, X., Sun, J., Qi, X. & Sun, J. Simultaneous modeling of car-following and lane-changing behaviors using deep learning. *Transportation research part C: emerging technologies* **104**, 287–304 (2019) .
- [41] Design and use paradigms for gazebo, an open-source multi-robot simulator. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2149-2154 (IEEE, 2004).

- [42] Chevalier-Boisvert, M., Golemo, F., Cao, Y., Mehta, B. & Paull, L. Duckietown environments for openai gym. <https://github.com/duckietown/gym-duckietown> (2018).
- [43] Tan, J. *et al.* Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332* (2018). Preprint at <https://arxiv.org/abs/1804.10332>.
- [44] Paull, L. *et al.* Duckietown: an open, inexpensive and flexible platform for autonomy education and research (2017). In IEEE International Conference on Robotics and Automation (ICRA) 1497-1504 (IEEE, 2017).
- [45] Ziegler, J. G. & Nichols, N. B. Optimum settings for automatic controllers. *Transactions of the American society of mechanical engineers* **64** (8), 759–765 (1942).
- [46] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8* (6), 679–698 (1986). <https://doi.org/10.1109/TPAMI.1986.4767851>.
- [47] Ballard, D. H. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition* **13** (2), 111–122 (1981).
- [48] Heikkila, J. & Silvén, O. A four-step camera calibration procedure with implicit image correction (1997). In Proceedings of IEEE computer society conference on computer vision and pattern recognition 1106-1112 (IEEE, 1997).
- [49] Censi, A. & Tipaldi, G. D. Lazy localization using the frozen-time smoother. In IEEE International Conference on Robotics and Automation 2778-2783 (IEEE, 2008).
- [50] Nelson, D. R., Barber, D. B., McLain, T. W. & Beard, R. W. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics* **23** (3), 519–529 (2007).
- [51] Quigley, M. *et al.* Ros: an open-source robot operating system. In ICRA workshop on open source software, vol. 3, 5 (IEEE, 2009).
- [52] Fujimoto, S., Hoof, H. & Meger, D. Addressing function approximation error in actor-critic methods. In International conference on machine learning 1587-1596 (PMLR, 2018).
- [53] Haarnoja, T. *et al.* Soft actor-critic algorithms and applications (2018). Preprint at <https://arxiv.org/abs/1812.05905>.

- [54] Meng, L., Gorbet, R. & Kulić, D. Memory-based deep reinforcement learning for pomdps (2021). In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 5619-5626 (IEEE, 2021).
- [55] Waltz, M. & Paulig, N. Rl dresden algorithm suite. [https://github.com/MarWaltz/TUD\\_RL](https://github.com/MarWaltz/TUD_RL) (2022).