Reinforcement Learning from Simulation to Real World Autonomous Driving using Digital Twin

Voogd, Kevin L.; Allamaa, Jean Pierre; Alonso-Mora, Javier; Son, Tong Duy

DOI
10.1016/j.ifacol.2023.10.1846

Publication date
2023

Document Version
Final published version

Published in
IFAC-PapersOnLine

Citation (APA)
Voogd, K. L., Allamaa, J. P., Alonso-Mora, J., & Son, T. D. (2023). Reinforcement Learning from Simulation to Real World Autonomous Driving using Digital Twin. *IFAC-PapersOnLine*, *56*(2), 1510-1515. https://doi.org/10.1016/j.ifacol.2023.10.1846

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 1510–1515

Reinforcement Learning from Simulation to Real World Autonomous Driving using Digital Twin ⋆

Kevin L. Voogd [*,**] **Jean Pierre Allamaa** [*]
Javier Alonso-Mora [**] **Tong Duy Son** [*]

[*] *Siemens Digital Industries Software, 3001 Leuven, Belgium (e-mail: {kevin.voogd, jean.pierre.allamaa, son.tong}@siemens.com)*
[**] *Faculty of Mechanical, Maritime and Materials Engineering, Technical University of Delft, 2628 CD Delft, The Netherlands (e-mail: j.alonsomora@tudelft.nl).*

Abstract: Reinforcement learning (RL) is a promising solution for autonomous vehicles to deal with complex and uncertain traffic environments. The RL training process is however expensive, unsafe, and time-consuming. Algorithms are often developed first in simulation and then transferred to the real-world, leading to a common sim2real challenge where performance decreases when the domain changes. In this paper, we propose a transfer learning process to minimize the gap by exploiting digital twin technology, relying on a systematic and simultaneous combination of virtual and real world data coming from vehicle dynamics and traffic scenarios. The model and testing environment is evolved from model, hardware to vehicle in the loop and proving ground testing stages, similar to standard development cycle in the automotive industry. In particular, we also integrate other transfer learning techniques such as domain randomization and adaptation in each stage. The simulation and real data are gradually incorporated to accelerate and make the transfer learning process more robust. The proposed RL methodology is applied to develop a path-following steering controller for an autonomous electric vehicle. After learning and deploying the real-time RL control policy on the vehicle, we obtained satisfactory and safe control performance already from the first deployment, demonstrating the advantages of the proposed digital twin based learning process.

Keywords: Learning and adaptation, autonomous vehicles, Sim2Real, reinforcement learning

1. INTRODUCTION

Research on autonomous vehicles (AVs) has made significant progress with recent advances in deep learning (DL), especially on the vehicle perception stack. While there have been some encouraging results and demonstrations, the application of DL on the vehicle planning and control stacks is still limited. Deep reinforcement learning (DRL) is an approach to generate control strategies in sequential processes, and capable of automatically learning and adapting from data, robustly against different operating conditions and tasks. This offers a more flexible and higher performance planning or control solution than traditional model-based control methods, which rely on a well-defined mathematical model of the system. Recent DRL breakthrough examples include AlphaStar (Arulkumaran et al., 2019), a model designed to play StarCraft II and end-to-end autonomous lane keeping (Kendall et al., 2019).

Despite the progress, further research and testing are crucial to realize the advantages of DRL, to be implementable in real-world within automotive industry standard (You et al., 2019). The main DRL challenge is a safe and efficient training process and testing environment: DRL training is expensive, time-consuming, and involves exploration of unsafe, risky situations. Training with the physical car is not legally possible in real traffic and is often limited to closed tracks. One may first train with collected human driving data and then transfer the controller to the physical world; however, the data misses out on critical scenarios that are needed to robustify the controller. The alternative training environment is simulation, where virtually generated data is cheap and fast, already labeled, and includes a large number of critical scenarios. Still, virtual data lacks true real-life properties and interactions, leading to a performance difference known as the sim2real transfer gap, when a policy is trained in simulation with a physical world deployment aim. This gap is caused by the simulation-optimization-bias where the controller exploits faults in the simulator and overestimates its performance compared to the target domain (Muratore et al., 2021). Other causes are model mismatch, noise, or actuation delays.

In this paper, we propose a DRL training and testing environment relying on Digital Twin (DT). DT is a virtual representation of a physical product or process, used across its development cycle to simulate and optimize the system's

⋆ This project was funded from the European Union's Horizon 2020 research and innovation programme under grant agreements No 956123 (FOCETA) and No 953348 (ELO-X). The work also benefited from the Flanders Innovation & Entrepreneurship – VLAIO funded project BECAREFUL.

10.1016/j.ifacol.2023.10.1846

performance and efficiency. In AVs, it comprises virtual models of vehicle dynamics, traffic scenarios and sensors. We exploit closed-loop DT which provides bi-directional connectivity between the physical and the virtual environment. In particular, the fidelity and complexity of models and environment are gradually evolved from model-in-the-loop (MiL) in simulation to hardware-in-the-loop (HiL) with physical vehicle embedded controller and actuation components, and vehicle-in-the-loop (ViL) proving ground testing with a real Siemens SimRod drive-by-wire car. The process provides a robust, high performance transfer learning, and is easier for prediction and tuning. Note that this is also known as V-cycle in industry, being adapted to DRL development purpose. Finally, we combine with domain adaptation and domain randomization techniques to enhance the transfer learning process. The transferred controller is deployed on the real vehicle successfully without fine-tuning in the target domain as shown in the abstract video at `https://youtu.be/RqIH9maKMxs`. The contributions of this paper are:

- a zero-shot transfer learning approach that combines the advantages of virtual training with real-world data. The DRL agent is robust to different paths and model uncertainties,
- a reduction in the sim2real gap for autonomous driving applications. The RL agent is trained using a high-fidelity (HF) vehicle dynamics simulator and traffic scenario simulator with domain randomization and adaptation,
- a deployable algorithm on a real-time operating system and a validation framework in MiL, HiL and ViL minimizing the overall testing effort and cost.

The paper is organized as follows. Section 2 summarizes related work on reinforcement learning for path following and sim2real methods. Section 3 reviews the background theory of RL, the vehicle model, and the transformations needed in domain adaptation. The experimental setup and the implementation details employed in this work are presented in Section 4.2, Section 5 provides a discussion on the results and concludes this work.

2. RELATED WORK

DRL and path following in autonomous driving: the first successful application of DRL in autonomous driving was achieved by Kendall et al. (2019), learning a control policy for lane-keeping from monocular images and training only in the physical world. Recent work by Alomari et al. (2021) claims to have developed a method that bridges the Sim2Real gap using a 3D vehicle dynamics simulator and parameter randomization, but the results are not validated on a real-time operating system. A similar study by Maramotti et al. (2022) focuses on a DRL planner using the single-track kinematic model and an additional neural network (NN) to simulate the state transition dynamics of the car. To speed up convergence, they pre-train the network with imitation learning and randomize the path and the vehicle's initial state. Similarly, Jiang et al. (2022) uses NNs to model the vehicle dynamics and constrains weights and activation functions of the DRL algorithm, turning it into a convex optimization problem. However, the computational time is significant, making it unfeasible for real-time application.

Sim2Real methods: Domain adaptation (DA) maps features from the source domain to the target domain, and *vice versa* or both, to a common latent space in an attempt to train the agent in a domain-independent framework. DA has been used to transform synthetic images or point clouds into realistic representations through generative adversarial networks (Xinlei Pan and Lu, 2017). Other researchers argue that data representation is the main source of the transfer gap and propose to transform the representation to lidar maps (Wang et al., 2019) or bird-eye views (Ng et al., 2020). Domain randomization (DR) is a method in which the parameters of the source domain are randomized so that it contains the target domain in its distribution. This method is effective when used with domain-specific knowledge and results in a robust control policy to model uncertainties as performed in Allamaa et al. (2022) to automatically tune the controller parameters. Randomization techniques have been applied to vehicle dynamics and physical parameters such as masses, friction, (Peng et al., 2018), trajectories, or random forces (Pouyanfar et al., 2019). It has also been performed on sensor data where DR techniques to alter poses, textures, dimensions, or colors (Andrychowicz et al., 2020). Lastly, system identification is used to identify properties of dynamical systems based on experimental measurements, which are later used to simulate the process more accurately. In addition, a Digital Twin is a HF multiphysics model that uses the available models and sensors to recreate in simulation its real life counterpart. By combining this model with real-time data, it is possible to accurately predict the vehicle's behavior (Hartmann and Van der Auweraer, 2020).

We propose a transfer learning approach to systematically combine all three techniques, allowing rapid and safe prototyping of DRL algorithms in real-world applications.

3. BACKGROUND

This section introduces the theory of RL, the vehicle kinematic model used in this work and lastly, the error frame transformation.

3.1 Reinforcement learning

Formally, RL problems are formulated as Markov decision processes (MDPs). Specifically in autonomous driving, RL is used to solve the MDP for the optimal driving policy. At every step, the MDP is composed by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the set of states and actions, respectively. In stochastic processes $\mathcal{P}(s_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the transition probability function of entering the state s_{t+1} from state s_t by taking the action a_t. Moreover, MDPs have the property that the conditional probability of a future state depends only on the present state. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, is the reward function that maps the state s_t in which the action a_t is taken and the resulting state s_{t+1} into a scalar value . The action is chosen based on a policy $\pi_\theta(a|s)$. In DRL, the policy is approximated with deep NNs with parameters θ. Given stochastic transition and policy functions, the objective is to maximize the return R_t over the trajectory, i.e. the expected reward:

$$J(\pi) = \int_\tau \mathcal{P}(\tau|\pi) R_t = \mathbb{E}\left[R_t\right]. \tag{1}$$

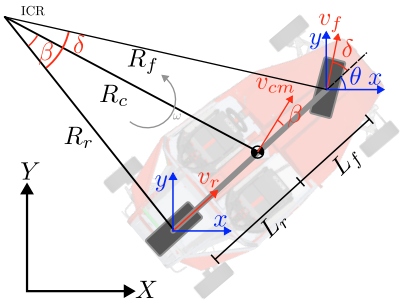

Fig. 1. Single-track model

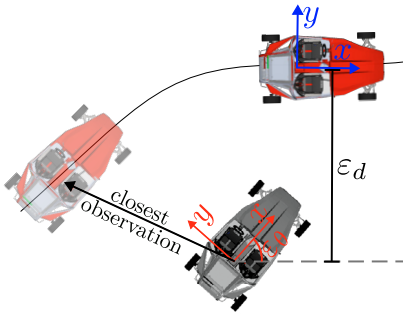

Fig. 2. Lateral deviation and yaw error calculation for the learning agent (gray). The deviation is computed w.r.t. the closest next observation (red).

The policy that maximizes the objective $J(\pi)$ is noted as π^*. In addition, there are two other functions: the state-value $V^\pi(a|s)$ function, which is the expected return of following the policy $\pi_\theta(a_t|s_t)$ from state s. The $Q^\pi(a|s)$ is the action-value function and maps the expected return of choosing an arbitrary action a in state s and then effectively follow the policy $\pi(\cdot|s)$. In this paper, we use the soft-actor critic (SAC) algorithm developed in Haarnoja et al. (2018), which is based on an actor-critic structure: the *actor* selects the action of the agent and the *critic* evaluates the action by approximating the Q_π-function. SAC includes entropy regularization in its objective function to encourage exploration, preventing early convergence to local minima. Additionally, this algorithm is off-policy meaning that the Q_π-function is learned from actions taken by a different policy than the current $\pi(a|s)$. Proximal Policy Optimization (PPO) was also tested in this work. However, it did not yield safe transferable policies, probably due to rapid convergence to a local minimum, and was therefore not a good candidate for our application.

3.2 Single-track model

The single-track bicycle model is a kinematic model of a four-wheeled vehicle (Fig. 1), in which the wheels at each axle are joined together. This model assumes a no-wheel slip condition. The length of the wheelbase is denoted as L, and the distances from the rear and front axle to the vehicle's center of gravity (CoG) are L_r and L_f, respectively. The vehicle's linear and angular velocities measured in a global reference frame are:

$$\begin{aligned} \dot{x}_{cm} &= v\cos(\theta+\beta) \\ \dot{y}_{cm} &= v\sin(\theta+\beta) \\ \dot{\theta} &= v/R_c, \end{aligned} \tag{2}$$

where θ is the heading of the chassis with respect to the global frame, $\beta = \arctan((L_r/L)\tan\delta)$ is the body slip angle. and the CoG's radius of rotation is $R_c = L/(\tan\delta\cos\beta)$. Inputs to the model are the steering angle δ and velocity v, controlling positions x_{cm}, y_{cm}.

Lateral and heading deviation: In this work, we use a buffer of recorded virtual and real-world trajectories to represent the centerline of the path to be driven. The heading and lateral deviations are calculated with respect to the closest next point in distance in the buffer. In Figure 2, the black and red vehicles represent the RL agent and the logged data respectively. The heading error (ε_θ) is calculated by computing the shortest difference between angles. The lateral deviation (ε_d) is calculated as:

$$[dx\ dy\ 1]^{\mathrm{T}} = \left({}^{E}_{W}H\right)^{-1}\left[{}^{W}x_S\ {}^{W}y_S\ 1\right]^{T}, \tag{3}$$

where S and E are the subscripts for the learning agent and the closest next observation respectively, W is the global frame, dx is the longitudinal offset, $dy = \varepsilon_d$ is the lateral deviation, and H is the homogeneous transformation matrix.

4. DRL TRAINING AND TRANSFER LEARNING

This section starts with a description of the training loop depicted in Figure 3 followed by a detailed explanation of the validation workflow. First, different trajectories are generated both synthetically in simulation (Simcenter Prescan) and in the physical world, and saved in the buffer. Training starts with the RL implementations (Raffin et al., 2021) with simulated data. When the performance stops improving, real-world trajectories are injected as the first step in DR to generalize on the true noise level and dynamic driving style. All collected data are transferred to a path following formulation, allowing us to set the observations in the error frame presented in Fig. 2. This enables a domain and reference invariant path following task, rendering it adaptable to both real and virtual domains. The episodes start with a random initialization of the environment and vehicle states, the second component of DR. The output of the DRL algorithm, the steering angle control action, is sent to the 15 DoF high fidelity (HF) vehicle dynamics simulator, the DT of the SimRod vehicle in Simcenter Amesim. The DT includes a number of identified parameters that are also randomized, to account for the sim2real gap. This third level of DR allows for even more generalization and robustification in the transfer learning approach. The performance during training is evaluated regularly every 2500 steps in 4 randomly selected scenarios and random initial conditions. The metric used to evaluate is the average timestep reward.

Data generation: virtual data is generated in the Simcenter Prescan, a traffic scenario simulator, at a frequency of 20 Hz with the same format as the real-world samples to facilitate the domain transition. The real-world samples are collected with the physical SimRod car in the company parking lot seen in Fig. 3, using a high accuracy dGPS.

4.1 Reinforcement learning training setup

State space: The NN inputs are processed sensor readings: the position in a global reference frame and inertial

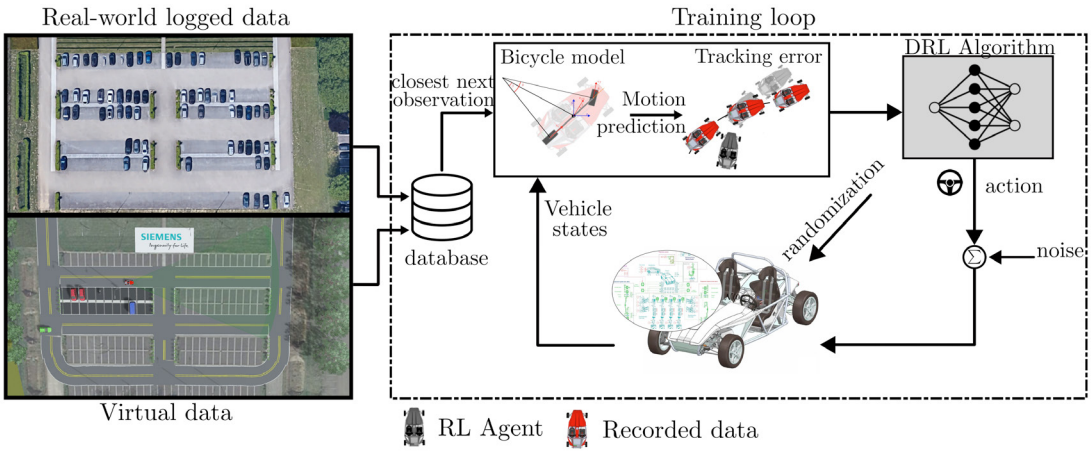

Fig. 3. Training of reinforcement learning policies using first synthetically generated data and a digital twin of the vehicle until performance settles. Then, real-world logged data is used. The predicted deviation is calculated with the single-track model. The initial states, the vehicle's physical parameters, and the control action are randomized.

measurements or virtually generated. These are the longitudinal speed of the vehicle v_x, the heading error ε_θ, the lateral deviation ε_d and their derivatives. The predicted lateral deviation is estimated with the single-track model for the next 10 timesteps, assuming constant speed and steering. More timesteps lead to a higher prediction inaccuracy, especially at early stages of training, and fewer timesteps may be short-sighted, hence the choice of 10 timesteps. In addition, the previous steering rate $\dot{\delta}$ and angle δ are inputted. By providing the NN with states in the error frame, we benefit from the possibility to generalize any reference path with any center of the coordinate system. This allows training independently of the domain and task to be performed.

Action space: The action generated by the policy network is the steering rate $\dot{\delta}$ saturated in the range $[-0.18, 0.18]$ rad/s. We opt for a generalized state-dependent exploration (gSDE), where the noise is dependent on the state of the car for the entire duration of the episode (Raffin et al., 2022). The steering rate is then integrated to obtain the steering angle. The longitudinal acceleration is computed with a PD controller.

Reward function: the reward function depends on the heading error ε_θ, the lateral deviation ε_d, and the steering rate $\dot{\delta}$. We propose to simultaneously optimize tracking and input by multiplying the individual components, and normalize them between 0 and 1 as:

$$r = \left(1 - \frac{|\varepsilon_\theta|}{\varepsilon_{\theta_{\max}}}\right)\left(1 - \frac{|\varepsilon_d|}{\varepsilon_{d_{\max}}}\right)\left(1 - |\dot{\delta}|\right). \tag{4}$$

Parameter randomization: We randomize the digital twin physical parameters to robustify against modeling errors and uncertainties such as changing road conditions, the number of passengers, or delays. Consequently, we randomize the mass, the location of the center of gravity, the length of the wheelbase, the suspension, and the stiffness of the tire. Values are drawn from a normal distribution. Every episode includes randomization of the initial deviation and orientation with respect to the path, to train the algorithm to overcome such uncertainties.

4.2 Implementation details and Experiments

The DRL model is trained on a laptop with 64 GB of RAM, an Intel Xeon W-11855M processor, and an NVIDIA RTX A4000 laptop graphics card. The model used for training has 16 inputs, 6 layers, and only one output, and also has a linear decay on the learning rate. The preprocessing steps and the NN are C code generated and deployed to the embedded platform dSPACE MicroAutobox III, running the real-time controller that commands the SimRod.

We train four different policies and then evaluate them in a standard V-cycle procedure satisfying the safety requirements: Model-in-the-loop (MiL), hardware-in-the-loop (HiL), and vehicle-in-the-Loop (ViL). The evaluated policies are SAC-ST-RW, SAC-HF-VD, and SAC-HF-RW, which abide by the notation: trained only with virtual data (VD), fine-tuned with real-world data (RW), single-track model (ST), and high-fidelity model (HF). The SAC-ST-RW is evaluated with the higher-fidelity model. MiL allows for safe and extensive verification and validation of the trained policies against possible edge cases, and actuator noise levels and provides initial performance metrics. This step enables a safe and cheaper transition to HiL and ViL levels. The learned policies are evaluated in real-world scenarios and are consistently evaluated at different initial deviations: $\{-1.25, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5\}$ meters but the vehicle's physical parameters are kept constant to compare the performance under equal conditions. ViL is carried out in a closed parking lot in Leuven, Belgium.

5. RESULTS AND DISCUSSION

In this section, we present the results of the training phase, as well as the evaluation of the MiL and ViL experiments. Figure 4 shows the average timestep reward of the SAC-HF-RW model, the best performing policy. The blue line shows the performance over time using only virtual data. Then, it is fine tuned with real-world data (red) as a component of the transfer learning methodology. There is a large decay (sim2real gap) when the data type is changed, showing the necessity of introducing such transfer learning logic, as the pre-trained model

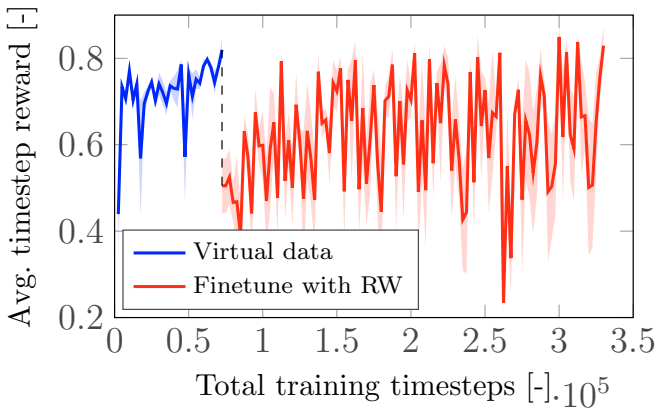

Fig. 4. Training results for SAC-HF agent: the agent is first trained with virtual data and the best model is further fine-tuned with real-world data.

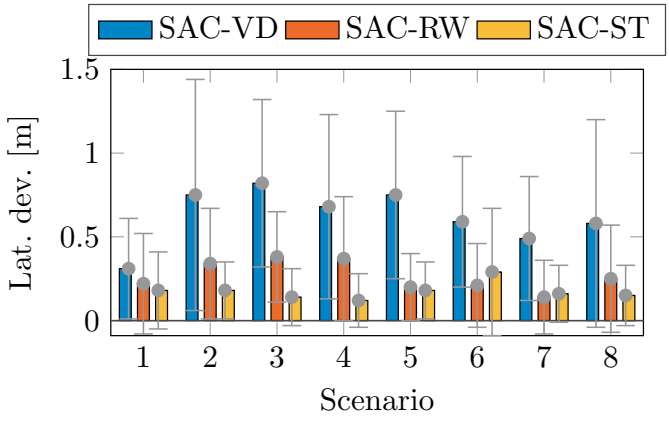

Fig. 5. MiL tracking performance of the DRL policies evaluated in eight different scenarios.

would have suffered from the sim2real gap. The model continues to train until settling in terms of performance. The model with the higher average return is evaluated in MiL and ViL. We observe a smaller variance during pre-training because the virtual scenarios are noise-free and simpler. These paths can be generated in millions and can speed up the pre-training phase, however, these are kinematic trajectories, which sometimes are not feasible for agents to track. Adding the real-world data afterward introduces dynamically possible curvature changes and state transitions. The amount of time required to perform 100000 timesteps with the HF model is 11.73 ± 0.35 hours. On the contrary, the kinematic ST only requires 42.47 ± 0.13 minutes for the same number of steps.

5.1 Model-in-the-Loop and Hardware-in-the-Loop

The results of the MiL experiments are shown in Figures 5 and 6. In general, the SAC-ST model performs the best with an average error of 17.5 cm for all trajectories. The SAC-HF-RW model error is 26.3 cm, and then SAC-HF-VD with 62.1 cm. Since training scenarios are performed only in 2D and not in 3D, the HF model adds unnecessary complexity. In other words, since in MiL the target domain is still a simulator with predefined parameters, the policy tends to compromise performance for robustness. This is also the reason for using DR, as the learning agent

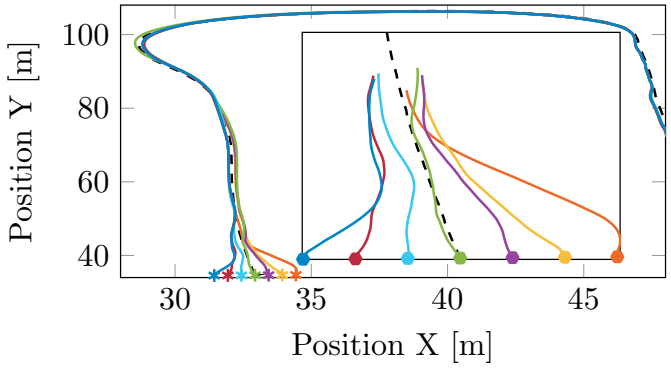

Fig. 6. MiL evaluations for the SAC-ST-RW agent with different initial deviations (in asterisk *)

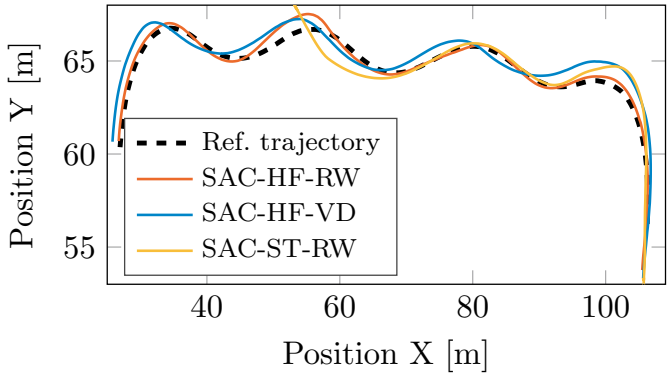

Fig. 7. ViL evaluation in the same path for all agents

tends to overtune in simulation, creating an SOB when transferring to the real-target domain. However, the ST performs worse than the HF model when dealing with a set of curves or higher-speed scenarios. The former is related to the fact that the executed steering angle is subjected to the assumption of a small steering motion and a large radius of curvature. We also noticed that using only virtual data with a HF model (SAC-HF-VD) is not enough for a performant transfer, hence the benefit of our proposed transfer learning approach.

5.2 Vehicle-in-the-Loop validation

As the main objective of this work is to transfer the policy efficiently and in zero-shot between the domains, we validate it in the actual vehicle. To our surprise, the SAC-ST-RW decreases its performance considerably when transferred to the real target. Hence, the benefit of the safe transfer learning methodology is proven, as the SAC-ST-RW does not account for a sim2real gap. The results are shown in Table 1 and Figure 7, pointing out that the best model is the policy trained with the HF model with an average deviation of 35 cm for all paths. On the contrary, the average error for the ST model increases to 49 cm; for SAC-HF-VD, the error is 52.5 cm. We also compare the transfer gap by computing the ratio between the performance in ViL and MiL. The SAC-ST-RW model deviates 4.75, 1.55, and 2.81 times more in the paths where ViL was tested. In contrast, the performance of SAC-HF-RW is not affected to the same extent by the sim2real gap. Specifically, we observe the following ratios: 1.60, 1.05, and 1.50. The HF model increases the tracking accuracy

Table 1. ViL results of the DRL policies tested in different scenarios

	Lateral deviation ($\mu \pm \sigma$) [m]			
	Path 3	Path 6	Path 7	Path 8
SAC-HF-RW	0.59+0.41	0.22+0.19	0.21+0.21	0.39+0.43
SAC-HF-VD	0.50+0.55	0.62+0.34	0.47+0.40	0.51+0.53
SAC-ST-RW	0.57+0.33	0.45+0.39	0.45+0.48	-

on average by 28.6% compared to the single-track model. Moreover, using real-world data to fine-tune the model results in 33.3% better transfer. Contrary to what has been reported by Truong et al. (2022), our results indicate that a higher-fidelity model is indeed necessary to minimize the reality gap. There was random locking of the vehicle's steering wheel in various trials; nevertheless, the agent was able to recover from the faulty states in the majority of cases and follow the path immediately afterward.

6. CONCLUSION

In this work, we develop a transfer learning strategy to efficiently train a DRL policy in simulation and deploy it in a real-time vehicle application. We show that standard approaches of training exclusively with virtual data or low-fidelity models are not sufficient to robustify the trained agent, even though they yield better performance in MiL. We combine state-of-the-art sim2real methods such as DR, DA, and HF with virtual and real-world data and show that they are all necessary components for safe transfer. The HF dynamics simulator allows efficient randomization of a large variety of parameters and correctly predicts the behavior of the vehicle under different conditions, robustifying the controller to real-world conditions and allowing a better zero-shot transfer. This work also focused on a safe and scalable approach to prototyping and developing algorithms for autonomous driving applications with physical testing in automotive industry standards. Finally, we validate our approach on a real-time path following control application in MiL, HiL, and ViL development stages.

REFERENCES

Allamaa, J.P., Patrinos, P., Van der Auweraer, H., and Son, T.D. (2022). Sim2real for autonomous vehicle control using executable digital twin. *IFAC-PapersOnLine*, 55(24), 385–391.

Alomari, K., Mendoza, R., Goehring, D., and Rojas, R. (2021). Path following with deep reinforcement learning for autonomous cars. In *Proceedings of the 2nd International Conference on Robotics, Computer Vision and Intelligent Systems - Volume 1: ROBOVIS,*, 173–181. INSTICC, SciTePress.

Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1), 3–20.

Arulkumaran, K., Cully, A., and Togelius, J. (2019). Alphastar: an evolutionary computation perspective. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1861–1870. PMLR.

Hartmann, D. and Van der Auweraer, H. (2020). Digital Twins.

Jiang, K., Hu, C., and Yan, F. (2022). Path-following control of autonomous ground vehicles based on input convex neural networks. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 236(13), 2806–2816.

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.M., Lam, V.D., Bewley, A., and Shah, A. (2019). Learning to drive in a day. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 8248–8254.

Maramotti, P., Capasso, A.P., Bacchiani, G., and Broggi, A. (2022). Tackling real-world autonomous driving using deep reinforcement learning. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, 1274–1281.

Muratore, F., Gienger, M., and Peters, J. (2021). Assessing transferability from simulation to reality for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4), 1172–1183.

Ng, M.H., Radia, K., Chen, J., Wang, D., Gog, I., and Gonzalez, J.E. (2020). BEV-Seg: bird's eye view semantic segmentation using geometry and semantic point cloud.

Peng, X.B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*.

Pouyanfar, S., Saleem, M., George, N., and Chen, S.C. (2019). ROADS: randomization for obstacle avoidance and driving in simulation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1267–1276. IEEE.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), 1–8.

Raffin, A., Kober, J., and Stulp, F. (2022). Smooth exploration for robotic reinforcement learning. In *5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, 1634–1644.

Truong, J., Rudolph, M., Yokoyama, N., Chernova, S., Batra, D., and Rai, A. (2022). Rethinking Sim2Real: Lower Fidelity Simulation Leads to Higher Sim2Real Transfer in Navigation. In *Conference on Robot Learning (CoRL)*.

Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., and Weinberger, K.Q. (2019). Pseudo-lidar from visual depth estimation: bridging the gap in 3D object detection for autonomous driving. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8437–8445.

Xinlei Pan, Yurong You, Z.W. and Lu, C. (2017). Virtual to real reinforcement learning for autonomous driving. In *Proceedings of the British Machine Vision Conference (BMVC)*, 11.1–11.13. BMVA Press.

You, C., Lu, J., Filev, D., and Tsiotras, P. (2019). Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems*, 114, 1–18.