Bypassing the Simulation-to-reality Gap: Online Reinforcement Learning using a Supervisor

Benjamin David Evans[1], Johannes Betz[3], Hongrui Zheng[2], Herman A. Engelbrecht[1],
Rahul Mangharam[2], and Hendrik W. Jordaan[1]

Abstract— Deep reinforcement learning (DRL) is a promising method to learn control policies for robots only from demonstration and experience. To cover the whole dynamic behaviour of the robot, DRL training is an active exploration process typically performed in simulation environments. Although this simulation training is cheap and fast, applying DRL algorithms to real-world settings is difficult. If agents are trained until they perform safely in simulation, transferring them to physical systems is difficult due to the sim-to-real gap caused by the difference between the simulation dynamics and the physical robot. In this paper, we present a method of online training a DRL agent to drive autonomously on a physical vehicle by using a model-based safety supervisor. Our solution uses a supervisory system to check if the action selected by the agent is safe or unsafe and ensure that a safe action is always implemented on the vehicle. With this, we can bypass the sim-to-real problem while training the DRL algorithm safely, quickly, and efficiently. We compare our method with conventional learning in simulation and on a physical vehicle. We provide a variety of real-world experiments where we train online a small-scale vehicle to drive autonomously with no prior simulation training. The evaluation results show that our method trains agents with improved sample efficiency while never crashing, and the trained agents demonstrate better driving performance than those trained in simulation.

I. INTRODUCTION

A. Motivation

Deep reinforcement learning (DRL) is a growing, popular method in autonomous system control [1]. Like humans that learn from experiences over time, DRL algorithms learn control mappings from sensor readings to planning commands using only observations from the environment and reward signals defined by the engineer. In contrast to humans who learn in the real world, DRL agents are usually trained in simulation. These simulation environments require accurate sensor and dynamics models to represent the robot and its surrounding environment. Unfortunately, the accuracy of simulation environments is limited to maintain good computation time, resulting in the sim-to-real gap when the simulation-trained DRL agent is transferred to a real-world system [2].

[1] B.D. Evans, H.A. Engelbrecht, and H.W. Jodaan are with the Department of Electrical and Electronic Engineering, Stellenbosch University, South Africa (e-mail: bdevans, hebecht, wjordaan@sun.ac.za)

[2] H. Zheng, and R. Mangharam are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, USA (e-mail: hongruiz, rahulm@seas.upenn.edu)

[3] J. Betz, is with the Professorship Autonomous Vehicle Systems, Technical University of Munich, Munich, Germany (e-mail: johannes.betz@tum.de)
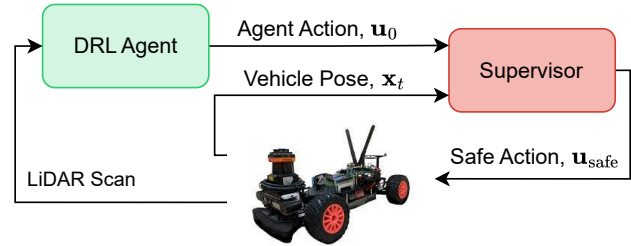
Fig. 1. A supervisor ensures the safety of a real-world vehicle during training a DRL agent. The supervisor uses the agent's action and the vehicle's pose to ensure that a safe action is selected.

It is desirable to train an agent directly on the robot, thus altogether avoiding the sim-to-real gap [3]. An inherent challenge in the online training of DRL algorithms on real-world robots is that DRL algorithms rely on crashing during training, meaning that training on a physical robot is very difficult or nearly impossible [4]. Crashing physical robots is expensive and a safety concern for the surrounding humans [5]. Therefore, being able to train DRL agents safely, crash-free onboard physical robots would enable the application of DRL agents to more physical platforms. Further, it can be expected that bypassing the sim-to-real gap will lead to improved DRL policies.

B. Contributions

We address the problem of training DRL agents (with no prior simulation training) on physical vehicles, thus ensuring their safety during the training process. Figure 1 shows our approach of using a supervisor to guarantee the vehicle's safety during the DRL agent training. The supervisory system uses a viability kernel (set of safe states) and vehicle model to check if the DRL agent's action is safe. If the DRL action is unsafe, a safe action from a pure pursuit controller is implemented. After training is completed, the supervisor is removed and the performance of the DRL agent is evaluated. This work has three main contributions:

- We combine a supervisory system with a DRL agent to guarantee crash-free training.
- We demonstrate that training an agent with a supervisor results in safe, robust, sample-efficient training of DRL agents in simulation and reality.
- We demonstrate agents trained onboard a real-world robot with the supervisor can effectively bypass the sim-to-real gap by outperforming an agent trained in simulation.

In §II, we study the literature on using DRL for autonomous vehicles. The methodology describes the F1Tenth platform in §III-A, the supervisory system in §III-B, the DRL formulation in III-C and the safety kernel generation in §III-D. We present the simulation and real-world evaluation results in §IV-A and §IV-B, respectively.

II. RELATED WORK

We discuss works related to DRL for autonomous vehicles, safe DRL training, and online DRL training.

DRL for autonomous vehicles: Many variations of DRL-based methods (model-based, model-free) have been implemented to derive control commands for autonomous vehicles from raw sensor inputs. The authors of [6], [7] used Deep Q-Learning (DQN) to learn steering manoeuvres for autonomous systems and [8], [9] used the soft-actor-critic (SAC) algorithm. Numerous deep learning studies are only evaluated in simulation because they are not practically feasible [10], [11], [12]. Of the DRL algorithms applied to physical systems, the dominant approach in the literature is to train the agents in simulation before transferring them to real vehicles [13], [14]. Evaluations show that DRL is an effective method of autonomous vehicle control, but the sim-to-real gap remains a challenge [15].

Safe DRL training: In [16] and [17], a risk-based approach is used to guarantee safety constraints during DRL training. While [16] uses a Monte Carlo tree search (MCTS) to reduce unsafe behaviours of the agent while training, [17] uses the estimation of trust region constraint to allow large update steps. Temporal logic specifications have also been used to enforce safety constraints during training [18], [19], [20]. A risk-based approach is poorly suited to autonomous vehicles since estimates of the risk cannot provide safety guarantees. Wang et al. [21] focuses on ensuring the legal safety of the vehicle by following traffic rules by using a safety layer based on control barrier functions. Control barrier functions and similar set theory techniques have been used in several safety-critical learning problems [22], but have been focused on applications where a safe setting can be assumed [23] or the dynamics can be simplified to linear (affine) equations [24].

Online DRL training: Training a DRL agent for autonomous driving is difficult since the only input regarding the map is an occupancy grid indicating if a block is open or filled. Kendal et al. [25] trained a DRL agent on a real-world vehicle using a safety driver (human intervention [26]) that decides to intervene if they think the car's position is unsafe. Bosello et al. [27] showed that a DRL algorithm on an autonomous vehicle could be trained by simply reversing the vehicle if it was near to crashing. These approaches demonstrate that online training for autonomous robots is a viable idea but is limited by a simplistic safety system. Musau et al. [28] used formal reachability theory to enable online training on a small-scale vehicle. Their method estimates future states of the vehicle in real-time resulting in it being computationally intensive and poorly suited to onboard hardware with limited computation.

In summary, safe online DRL training is a growing field that requires further investigation to explore how DRL agents can be trained onboard real-world robots while guaranteeing safety at the same time. Viable approaches should use the track occupancy grid to determine when the vehicle is on the edge of safety and only then intervene.

III. METHODOLOGY

A. F1Tenth Platform

F1Tenth racing cars are 1/10th the size of real F1 vehicles and are used as a test-bed for autonomous algorithms [29]. The platform focuses on safe algorithms that run autonomously onboard the vehicle. The cars are equipped with a LiDAR scanner for sensing the environment, an NVIDIA Jetson NX as the main computation platform, a variable electronic speed controller (VESC) and drive motor to move the vehicle forwards, and a servo motor to steer the front wheels. The vehicle uses the ROS2 middleware for the sensors, software components and control signals to communicate with each other.

Problem: We approach the problem of training a DRL agent to drive a F1Tenth vehicle autonomously around a provided race track. The task of planning is to use the onboard sensor measurements, LiDAR scan and odometry (estimated using a particle filter [30]) to calculate an optimal steering angle δ and velocity v that results in the vehicle driving around the track. Training a DRL agent means randomly initializing a policy (neural network), then using the policy to collect experience, and using the collected samples to adjust the policy parameters until the agent can drive around the track.

Vehicle Model: The vehicle is a controlled discrete-time system such that $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u})$. The vehicle state at the current timestep $\mathbf{x}_k$, comprises the vehicle location in the x and y directions and the vehicle orientation, such that $\mathbf{x}_k = [X, Y, \theta]$. The vehicle control $\mathbf{u}$ consists of a steering angle δ, and velocity v, such that $\mathbf{u} = [\delta, v]$. In our experiments, the vehicle speed is kept constant. State updates are performed using the single-track vehicle model [31]. The additional state variables in the single-track model (e.g. slip angle) are set to 0 before each update.

B. Supervisory Safety System

In contrast to solutions that train DRL agents in simulation and transfer them to physical vehicles [13], we present a safety supervisor that enables the training of DRL agents onboard the physical vehicle. We present the training architecture followed by a description of the supervisor operation.

Supervisory Training Architecture: The training architecture, shown in Figure 1, uses the supervisor to monitor the agent and ensure that only safe actions are implemented on the vehicle. Safe actions are those that do not lead the vehicle to crash into the boundary and are recursively feasible, i.e. after taking a safe action; another safe action will definitely exist. When training is complete, the agent is tested by removing the supervisor to demonstrate that the agent has learned to drive safely around the track.

Supervisor Operation: Figure 2 shows how the supervisor fulfils its role of ensuring that only safe actions are implemented on the vehicle through a three-step process of (1) using the current state and action to calculate the next state, (2) checking if the next state is safe, and (3) if unsafe, selecting a safe action.

The supervisor checks if an action $\mathbf{u}_0$ is safe by using the current vehicle pose $\mathbf{x}_t$ and a dynamics model to simulate the next state where the vehicle will be after the planning timestep. A kernel of all the possible states (positions and orientation) of the vehicle on the map is divided into the subsets of safe states $\mathcal{X}_{\text{safe}}$ and unsafe states through a process described in Section III-D. The next state is evaluated for safety by checking if it is in the subset of safe states. If the next state is safe, then the agent action can be implemented; otherwise, a safe action must be selected.

Selecting a safe action $\mathbf{u}_{\text{safe}}$ is done using a pure pursuit controller [32] that uses the single-track model of the car to calculate a steering angle that follows the centerline of the race track [33]. Using the pure pursuit controller in this way ensures that the vehicle will always move towards the center of the track where it is safe, away from the potential danger of the track boundaries.

C. Supervisory Reinforcement Learning

Reinforcement learning problems are modelled as Markov Decision Processes (MDPs) having a state space, action space, reward signal and transition probability. During training, the agent, consisting of a neural network, receives a state and selects an action that is implemented and a new state and reward are returned to the agent. The agent's experience of states, actions, next states and rewards is stored in a buffer and used to update the neural network parameters to select actions that maximise the reward signal. We use the twin-delayed-deep-deterministic-policy-gradient (TD3) algorithm [34] to train our agents to select continuous control actions.

The DRL agent uses neural networks with two fully connected hidden layers of 100 neurons each and the *ReLu* activation function. The agent's state vector (input) consists of 20 evenly sliced beams from the LiDAR scan scaled from the LiDAR beam range of 10 m to the range [0, 1]. The output is the steering angle, scaled from [-1, 1] (realized

Fig. 2. The supervisor ensures safety by simulating the next state, checking if the resulting state is safe and if unsafe, then selecting a safe action.

using the *tanh* activation function) to the steering angle range of 0.4 rad. The planners operate at a frequency of 10 Hz.

Training Reformulation: In conventional RL, an episode is an ordered set (or trajectory) of state, action, reward, and done tuples, from an initial state to a terminal state (crashing or completing a lap). Using the supervisor, the agent never crashes and always completes laps. Additionally, if the supervisor intervenes, a different action is implemented on the vehicle to that which the agent selected, breaking the link between state, action and next state.
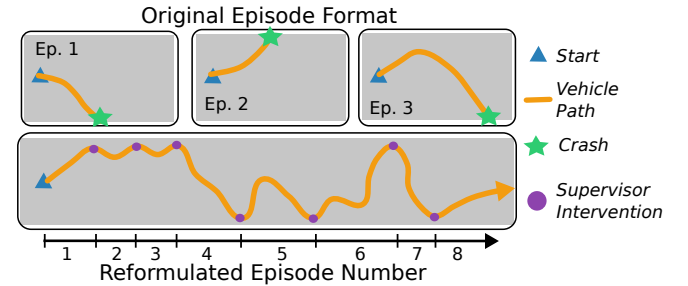
Fig. 3. Example vehicle paths comparing the original episode format of crashing and resetting against the reformulated episodes of the supervisor intervening.

Using the supervisor, we define an episode to run from the initial state until the supervisor intervenes. When the supervisor intervenes, it is recorded as a terminal state, and the supervisor gives the agent a penalty of -1. This penalty for unsafe actions is the only reward used by our method. Figure 3 shows how the definition of an episode has been changed from conventionally requiring many episodes with resetting the vehicle to shorter episodes running while the supervisor does not intervene.

D. Safety Kernel Generation

The supervisor uses a list (or kernel) of recursively safe states to ensure vehicle safety. As previously mentioned, safe states are defined as states that do not lead to the vehicle crashing into the boundary and are recursively feasible, meaning that every safe state has an action that leads to another safe state. The formulation and generation of the kernel of safe states are based on the work by Liniger et al. [35].

The state space $\mathcal{X}$ is discretized into a countable number of states $\mathcal{X}_h$. The track map is split into a finite number of blocks by gridding the map with a uniform grid with a resolution of 40 blocks per meter. The orientation angle θ is split into 41 even angle segments. The control space U, consisting of the steering angle range, is split into 9 evenly spaced control modes. The discrete states are used to formulate the dynamics as a difference inclusion where the next state $\mathbf{x}_{k+1}$ is in the set of possible next states, $\mathbf{x}_{k+1} \in F(\mathbf{x}_k)$. For a given state, the set of all possible next states is written as $F(\mathbf{x}_k) = \{f(\mathbf{x}_k, \delta) \mid \delta \in [-\delta_{\max}, \delta_{\max}]\}$.

The kernel of safe states $\mathcal{X}_{\text{safe}}$ is a subset of the discrete state space $\mathcal{X}_h$, for which there exists a safe action. The
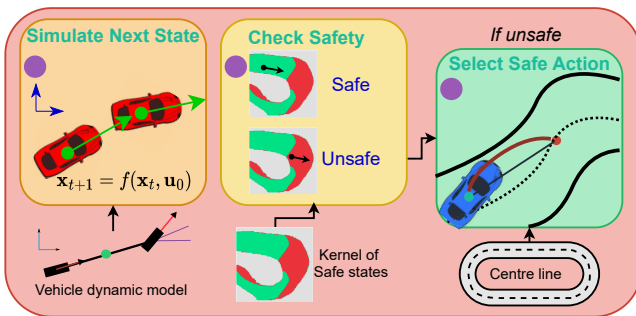
kernel is calculated using the recursive viability kernel algorithm,

$$K^0 = K_{\text{track}}$$
$$K^{i+1} = \{\mathbf{x}_h \in K^i \mid \forall\ F(\mathbf{x}_h) \cap K^i \neq \varnothing\}. \tag{1}$$
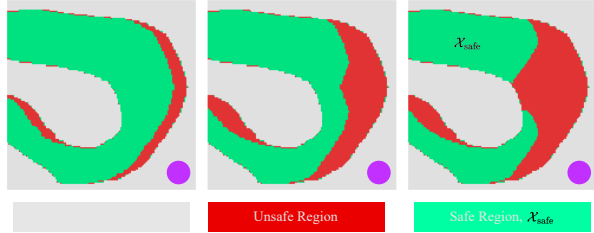
Fig. 4. Three stages of kernel growth for a corner on a race track with the vehicle pointing towards the right. Note that the kernel shape (and safe region) depends on the vehicle orientation.

The viability kernel algorithm in Equation 1 generates a set of states for which there recursively exists an action that causes the vehicle to remain within the kernel. The algorithm's safe set is initialized (K^0) to all the states on the drive-able area of the race track being safe, K_{track}. The algorithm then recursively generates smaller safe sets by looping through each safe state from the previous iteration and including only states for which there exists an action that leads to another safe state. Formally, this process is defined as selecting states for which the intersection of the next states and the kernel is not equal to the empty set. This process results in a 3-dimensional kernel of recursively feasible safe states, $\mathcal{X}_{\text{safe}}$. Figure 4 shows how the kernel grows inwards from the track boundaries until all remaining states are safe. While the kernel shape depends on the vehicle orientation, Figure 4 visualises the kernel with the vehicle orientation pointing towards the right.

IV. EVALUATION

We compare training using a supervisor against the conventional learning baseline in simulation. Then, online learning onboard a physical vehicle is compared to an agent trained in simulation and tested on a real-world platform.

Baseline: The baseline agent is trained with conventional DRL, where it crashes and is reset to a starting position. The baseline agent reward has a punishment of -1 for crashing, a reward of 1 for completing a lap and a shaped reward relative to the centerline progress (same as [13]). The shaped reward is calculated as, $r_t = (p_t - p_{t-1})/p_{\text{total}}$, where p_t is the centerline progress at timestep t, scaled according to the total centerline length p_{total}. For completed laps, the baseline agent receives a reward of 2, where 1 is the sum of intermediate progress rewards and 1 is for lap completion.

A. Simulation Tests

Conventional and online learning are compared in the open-source F1Tenth simulator [36] on scaled F1 race tracks. The experiments are repeated three times using different random seeds and the average used, and for each repetition,

20 test laps are completed. The simulation results use a constant speed of 2 m/s. Table I shows the shape of the AUT, MCO, GBR, and ESP tracks (from [27]), with the mean times used for the lap time normalisation. The maps come from Austria, Spain, England, and Monaco.

Track	AUT	ESP	GBR	MCO
Image				
Mean lap time (s)	46.7	116.4	100.2	86.6

TABLE I

TRACK IMAGES AND MEAN LAP TIMES FOR THE SCALED MAPS.

Training Comparison: Figure 5 shows the average episode rewards earned during the training of the baseline agents for 40,000 steps (left) and the online agents for 6,000 steps (right). The conventional agents start with earning a reward near -1, indicating they crash quickly. After around 20k steps, the average reward converges to between 1 and 2. The online agents start earning large negative rewards of around -100 or the AUT track and -250 for the other tracks, indicating that the supervisor intervenes a lot at the beginning of training. After around 4k training steps the agents all receive near to 0 reward indicating that the supervisor seldom intervenes.
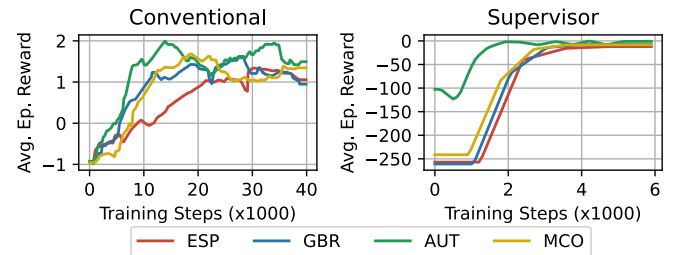
Fig. 5. Rewards earned during training of the baseline (left) and online (right) agents on the AUT, MCO, GBR and ESP maps in simulation.

Therefore, the simulated training comparison demonstrates that online training is more sample efficient than conventional training requiring only 6k training steps.

Performance Comparison: We study the performance of the baseline and online agents by considering the normalised lap times and success rates.
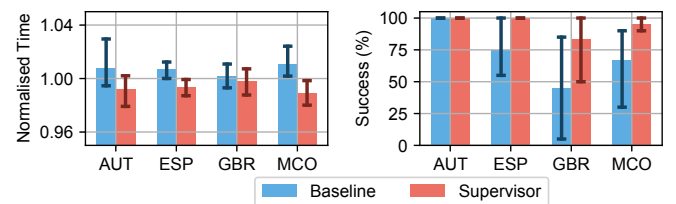
Fig. 6. Lap times and total curvature of the baseline and online agents compared to a pure pursuit planner.
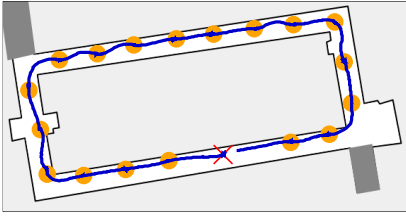
Fig. 7. Training lap (starting at the red cross) of the safety agent (blue line) with the locations where the agent stopped to train (yellow dots).

Fig. 8. Comparison of steering actions selected by the agent (green) and those implemented by the safety system (blue) during training of the safety agent in simulation in environment 1.

Figure 6 shows bar plots of the normalised lap times and success rates of the baseline and online agents. The error bars represent the minimum and maximum values from the three repetitions. The lap times are normalised by dividing them by the mean times shown in Table I. The success rate is the percentage of the test laps that were completed without the vehicle colliding. The agents trained with the supervisor achieved lower normalised lap times and higher success rates on all maps. For example, on the MCO track, the agent trained with the supervisor achieves a 95% average completion rate, while the conventionally trained agent achieves only 70%. The simulation results indicate that training with a supervisor results in faster lap times with higher success rates.

B. Real-world Tests

The supervisor is used to train a DRL agent, with no a priori knowledge or training, onboard a vehicle to drive around a track autonomously. The vehicle is trained by completing two laps in environment 1 (around 800 steps), driving at a constant speed of 2 m/s. Figure 7 shows the first training lap of a randomly initialised agent being trained online using the supervisor. The trajectory shows the agent's squiggles as it veers to one side and then to the other. Due to the computation burden of training the agent, the vehicle collects 20 steps of experience and then stops to train before continuing to collect more data (shown by yellow dots). The entire training process of driving two laps while stopping to train takes around 10 minutes.

Supervisor Effect: A graph comparing the steering angles selected by the agent (green) and the safe actions implemented by the supervisor (blue) is shown in Figure 8. At the beginning of the training, the agent rarely selects safe actions. As the training progresses, the agent selects more safe actions, and towards the end, the agent rarely selects an unsafe action. This result shows the supervisor's behaviour in preventing the agent from taking unsafe actions during the initial stage of training, and how, as the agent is trained, it learns to select safe actions without requiring the supervisor.

Online Training Rewards: We investigate the rewards earned by the agent during training with the supervisory system. We use the sum of the reward achieved every 20 steps (the interval of data collection between the agent stopping to train) as the metric to measure the online training performance. The worst reward is -20 if the supervisor
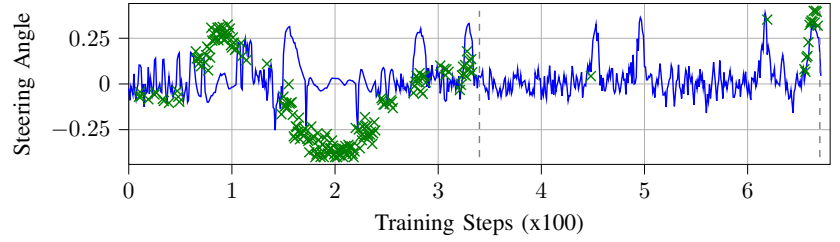
intervenes at every step, and the maximum reward is 0 if the supervisor never intervenes.
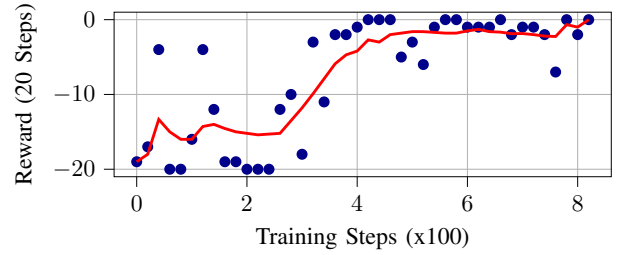
Fig. 9. Training rewards per 20 steps for safety agent trained on the physical vehicle (blue dots) with moving average (red).

Figure 9 shows a graph of the sum of rewards achieved every 20 steps by the agent trained online the physical vehicle. The graph shows that in the beginning, the agent receives low rewards; as time progresses, the agent receives higher rewards. After around only 400 steps, the agent displays a significant improvement, which corresponds to the graph of safe steering actions in Figure 8 showing that the agent requires less intervention between 300-400 training steps. This result shows that our method of using a supervisor is effective for training a DRL agent onboard a real-world vehicle in only 800 training steps.

Quantitative Analysis: We compare a DRL agent trained online a physical vehicle with the safety system against a baseline agent, trained offline in a simulator and then transferred to the vehicle. The baseline agent is trained in simulation on the environment 1 map for 30,000 steps.

Table II presents the quantitative results of the offline (baseline), and online (supervisor) trained DRL agents in two different environments with the metrics of distance travelled, lap time, absolute mean steering and curvature. The agent trained with the supervisor generally leads to a lower mean steering angle and lower total curvature of the trajectories, resulting in lower distance travelled and lower corresponding lap times than the baseline agents. For example, on the physical vehicle driving in environment 1 (shown in Figure 10), the baseline agent travelled 65.0 m, while the supervisory agent travelled only 59.8 m, which is 5.2 m shorter. The average steering angle for the supervisory agent was 0.03 radians, compared to the mean steering angle for the baseline of 0.3 radians. The baseline total curvature

	Environment 1				Environment 2			
	Simulation		Reality		Simulation		Reality	
	Baseline	Supervisor	Baseline	Supervisor	Baseline	Supervisor	Baseline	Supervisor
Distance Driven in m	65.0	**59.8**	65.68	**61.6**	17.0	**15.6**	18.8	**17.3**
Lap-time in s	32.8	**31.1**	35.5	**32.5**	12.9	**11.1**	12.8	**10.1**
Mean Steering Angle in rad	0.30	**0.03**	0.22	**0.06**	0.36	**0.11**	0.31	**0.09**
Total Curvature in m^{-1}	274.6	**34.2**	207.5	**86.3**	119.1	**44.9**	86.6	**49.3**

TABLE II

QUANTITATIVE COMPARISON OF ONLINE AND OFFLINE TRAINED DRL AGENTS IN TWO DIFFERENT ENVIRONMENTS.

was significantly more (207.5) than the supervisory agent's (86.3).

This result demonstrates that training agents with the supervisor outperforms conventionally trained DRL agents on real-world vehicles with smoother steering actions. Although the supervisory agent also performs worse in reality compared to simulation, training the DRL agent on the real car shows a definite improvement in the performance of the physical vehicle compared to the baseline.

Qualitative Analysis: Figure 10 shows the real-world trajectories for the baseline agent trained in simulation (red) and our method trained onboard the vehicle (blue). In the left image on environment 1 (training track), the baseline selects a squiggly, un-smooth path, regularly coming close to the track boundaries and almost crashing. In contrast, the DRL agent trained onboard the vehicle using the supervisor has a much smoother trajectory, driving in a line through straights and smoothly turning corners.
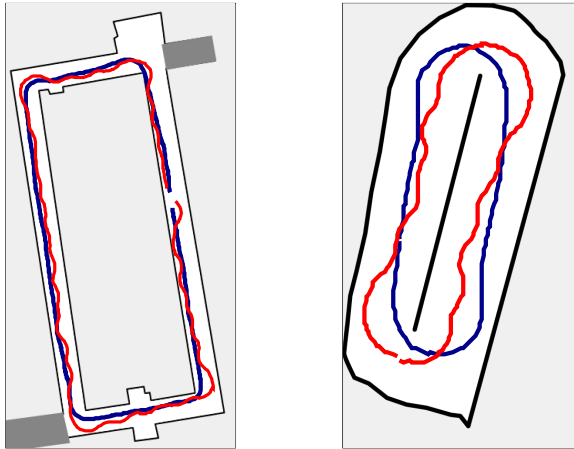
Fig. 10.   Comparing the test trajectories of the baseline (red) and online (blue) agents on environment 1 (left) and environment 2 (right)

Robustness: A crucial aspect of DRL agents is their ability to learn general policies that can be transferred to other environments. Both the agents trained in simulation and on the physical vehicle are tested on the track they were trained on (environment 1) and a different test track (environment 2, right in Figure 10). Due to the reduced size of the track, the speed was reduced to 1.5 m/s. The first observation is that both the baseline and supervisory

agents can complete laps on a different track to the one that they were trained on, highlighting the advantage of the flexibility and adequate generalization of DRL agents. The right image in Figure 10 shows that the trajectories followed in environment 2 display a similar pattern to that of environment 1. The supervisory agent takes a smoother path and swerves less than the baseline. This outcome is reinforced by the quantitative results in Table II, which show that the agent trained with the supervisor achieves a shorter lap time (1.5 s different), with a lower mean steering angle (0.09 versus 0.31) and less total curvature (49.3 versus 86.6) than the baseline planner. Therefore, we conclude that training a DRL agent onboard with a supervisor results in more general behaviour, as demonstrated by improving performance on a different track.

V. CONCLUSION

This paper presented a supervisory safety system capable of training a DRL agent for autonomous driving online on the vehicle car. The supervisor ensures vehicle safety by checking if the DRL agent's action is safe or unsafe, using a pure pursuit planner to select a safe action. We did two evaluations to prove the algorithm's robustness, once in simulation and once on a physical real-world vehicle. The evaluation in simulation demonstrated that using the supervisor to train agents results in lower lap times and higher success rates while requiring fewer training steps. The real-world test demonstrated that the supervisory system is effective for safely training a randomly initialized agent onboard a physical vehicle. The autonomous vehicle demonstrated a safely driven path in the given environment. The results showed that the agent trained online with the supervisor performed better than the agent trained purely in simulation by driving a shorter path around the track, and selecting a smoother path than the baseline without requiring additional measures (such as reward hacking in [27] or action regularisation in [13]). The agent trained with the supervisor could transfer to an environment unseen during training where it outperformed the conventionally trained agent. These results demonstrate that our method effectively bypasses the sim-to-real gap by training agents onboard real-world vehicles.

Future work should address expanding safe learning on-board physical robots to other physical platforms such as

UAV control and high-speed autonomous racing. This task requires extending the principle of formulating a supervisor and a safety policy to operate the robotic system at its performance limits and include more control actions, e.g. velocity, acceleration, and various vehicle dynamics parameters.

REFERENCES

[1] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016.

[2] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2020.

[3] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," *arXiv preprint arXiv:2004.12570*, 2020.

[4] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Mach. Learn.*, vol. 110, no. 9, pp. 2419–2468, Sep. 2021.

[5] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[6] K. Wu, H. Wang, M. Abolfazli Esfahani, and S. Yuan, "Bnd*-ddqn: Learn to steer autonomously through deep reinforcement learning," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 2, pp. 249–261, 2021.

[7] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[8] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb. 2022.

[9] S. Kuutti, R. Bowden, H. Joshi, R. d. Temple, and S. Fallah, "End-to-end reinforcement learning for autonomous longitudinal control using advantage actor critic with temporal context," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, Oct. 2019.

[10] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, "Superhuman performance in gran turismo sport using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.

[11] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2070–2075.

[12] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, "High-speed autonomous drifting with deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1247–1254, 2020.

[13] A. Brunnbauer, L. Berducci, A. Brandstátter, M. Lechner, R. Hasani, D. Rus, and R. Grosu, "Latent imagination facilitates zero-shot transfer in autonomous racing," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7513–7520.

[14] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7262–7269, 2021.

[15] E. Chisari, A. Liniger, A. Rupenyan, L. Van Gool, and J. Lygeros, "Learning from simulation, racing in reality," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8046–8052.

[16] S. Mo, X. Pei, and C. Wu, "Safe reinforcement learning for autonomous vehicle using monte carlo tree search," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6766–6773, Jul. 2022.

[17] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–7.

[18] M. Cai, E. Aasi, C. Belta, and C.-I. Vasile, "Overcoming exploration: Deep reinforcement learning in complex environments from temporal logic specifications," *arXiv preprint arXiv:2201.12231*, 2022.

[19] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11797

[20] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3834–3839.

[21] X. Wang, "Ensuring safety of learning-based motion planners using control barrier functions," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4773–4780, Apr. 2022.

[22] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[23] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[24] Z. Li, U. Kalabić, and T. Chu, "Safe reinforcement learning: Learning with supervision using a constraint-admissible set," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 6390–6395.

[25] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019. [Online]. Available: https://doi.org/10.1109/icra.2019.8793742

[26] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 2067–2069.

[27] M. Bosello, R. Tse, and G. Pau, "Train in austria, race in montecarlo: Generalized rl for cross-track f1 tenth lidar-based races," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2022, pp. 290–298.

[28] P. Musau, N. Hamilton, D. M. Lopez, P. Robinette, and T. T. Johnson, "On using real-time reachability for the safety assurance of machine learning controllers," in *2022 IEEE International Conference on Assured Autonomy (ICAA)*. IEEE, 2022, pp. 1–10.

[29] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.

[30] C. Walsh and S. Karaman, "Cddt: Fast approximate 2d ray casting for accelerated localization," vol. abs/1705.01167, 2017. [Online]. Available: http://arxiv.org/abs/1705.01167

[31] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.

[32] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[33] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," *Vehicle System Dynamics*, vol. 58, no. 10, pp. 1497–1527, Jun. 2019. [Online]. Available: https://doi.org/10.1080/00423114.2019.1631455

[34] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[35] A. Liniger and J. Lygeros, "Real-time control for autonomous racing based on viability theory," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, pp. 464–478, 2017.

[36] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proceedings of Machine Learning Research*, vol. 123, 2020.