# CommonRoad-RL: A Configurable Reinforcement Learning Environment for Motion Planning of Autonomous Vehicles

Xiao Wang, Hanna Krasowski, and Matthias Althoff

***Abstract*— Reinforcement learning (RL) methods have gained popularity in the field of motion planning for autonomous vehicles due to their success in robotics and computer games. However, no existing work enables researchers to conveniently compare different underlying the Markov decision processes (MDPs). To address this issue, we present CommonRoad-RL—an open-source toolbox to train and evaluate RL-based motion planners for autonomous vehicles. Configurability, modularity, and stability of CommonRoad-RL simplify comparing different MDPs. This is demonstrated by comparing agents trained with different rewards, action spaces, and vehicle models on a real-world highway dataset. Our toolbox is available at commonroad.in.tum.de.**

## I. Introduction

Reinforcement learning (RL) approaches are increasingly being used to handle motion planning for autonomous driving. In contrast to human-engineered motion planners, RL algorithms learn the optimal behavior through rewards during driving. In addition, complex tasks can be solved using complex function approximators, such as neural networks, to represent an agent. Researchers using RL for motion planning typically create environment models specifically tailored to the current research question. As a consequence, many variations of the Markov decision processes (MDPs), i.e., various action and state spaces, reward definitions, and driving scenarios, exist in the literature [1], and because of the effort and time required to implement several variations, these processes are rarely compared with each other.

Besides varying MDPs, traffic data used in simulated RL environments can originate from different sources, e.g., driver models or real traffic data. To compare different RL approaches, traffic data must be integrated in a deterministic way. The CommonRoad benchmark suite [2] offers a large number of deterministic benchmark scenarios, either constructed from real traffic or by running the traffic simulator SUMO [3], [4] in a deterministic way. In addition, all scenario data are well-defined to easily extract meaningful observations compared to extracting raw sensor data.

To effortlessly benchmark different MDPs of RL problems for motion planning in autonomous driving, we provide a configurable and deterministic RL environment. This RL environment is automatically generated according to user specifications using our novel toolbox CommonRoad-RL, which profits the following benefits:

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
xiao.wang@tum.de, hanna.krasowski@tum.de, althoff@in.tum.de

- it provides comprehensive definitions of state spaces, action spaces, and rewards for RL agents;
- it automatically creates a training and evaluation environment according to a configuration file defined by users;
- it enables users to quickly test state-of-the-art RL algorithms on real-world traffic scenarios, identifying the ideal choice of an MDP and reducing potential implementation errors;
- it offers a unified interface for several powerful motion planning tools, such as an efficient drivability checker [5], a map converter [6], a critical-scenario factory [7], and a scenario designer [8];
- it is provided as open-source code and works with all scenarios of the CommonRoad benchmark suite [2].

The remainder of this paper is structured as follows: Section II discusses related tools. Section III describes CommonRoad-RL and its features, and Section IV showcases our tool using recorded highway data. We conclude and give an outlook on future work in Section V.

## II. Related Work

Multiple open-source tools are available to train a driving agent. Driving simulators [9]–[12] require an interface to conduct RL, whereas RL environments [13]–[15] can be directly connected to RL algorithms, albeit often only supporting simple traffic simulations. We limit our review to the most popular tools and refer interested readers to a more comprehensive review [16]. An overview of the reviewed tools is provided in Tab. I.

### A. Autonomous Driving Simulators

CARLA [9] is an autonomous driving simulator with realistic driving environments, including traffic signs and lights as well as a variety of sensors. There are open-source implementations to include CARLA in an RL environment, and an open-source interface to import real-world traffic data [17]. LGSVL [10] is a simulator similar to CARLA that also includes realistic models of driving environments and many different sensors. Furthermore, an RL environment is provided that features continuous visual observation and continuous actions to a realistic vehicle. TORCS [12] is a racing simulator used for gaming and research. However, TORCS only provides racing tracks and cannot be used to simulate a realistic road environment. Automotive Simulator [11] is a simulator for autonomous driving written in Julia. It can import real-world road networks, driver models, and the NGSIM dataset [18]. Available RL interfaces of CARLA,

TABLE I: Related tools and their supported features.

| Tool | RL environment | Realistic scenarios | Real traffic data | Configurable MDP | Documentation |
|---|---|---|---|---|---|
| **CARLA** [9] | ◐ | ✔ | ✔ | ✖ | ✔ |
| **LGSVL** [10] | ✔ | ✔ | ✖ | ✖ | ✔ |
| **TORCS** [12] | ◐ | ✖ | ✖ | ✖ | ◐ |
| **Automotive Simulator** [11] | ✖ | ✔ | ✔ | ✖ | ✖ |
| **BARK** [13] | ✔ | ✔ | ✔ | ◐ | ◐ |
| **highway-env** [14] | ✔ | ◐ | ✖ | ◐ | ✔ |
| **OpenAI gym** [15] | ✔ | ✖ | ✖ | ✖ | ◐ |
| **CommonRoad-RL** | ✔ | ✔ | ✔ | ✔ | ✔ |

LGSVL, and TORCS do not feature a configurable observation and action space and, thus, are not easily adaptable. For Automotive Simulator, there is no explicit RL environment implemented. Furthermore, real traffic data cannot be imported into LGSVL and TORCS.

### B. RL Training Environments

BARK [13] is a simulation framework for behavior benchmarking of autonomous driving. It supports real traffic data, driver models, and multi-agent RL. The provided scenarios are highway, merging, and intersection scenarios. Another open-source RL environment is highway-env [14], which models ten different types of scenarios (e.g., highway, roundabout, and intersection scenarios) as well as supports a modular state and action space. While a multi-agent setting is possible, traffic participants other than cars are not supported, and road networks cannot be imported. OpenAI gym [15] defines a common interface for RL environments, which can be easily combined with open-source RL algorithms. In addition, OpenAI gym contains a collection of different RL environments; the most appropriate one for autonomous driving is the CarRacing environment. However, in CarRacing, there are no traffic participants, and a task cannot be easily adapted to realistic roads.

To summarize, among all existing tools, only BARK and highway-env support configurable MDPs. Yet, only a few types of states and actions are supported by them in contrast to CommonRoad-RL.

## III. CommonRoad-RL

### A. Overview

CommonRoad-RL enables motion planning for autonomous driving in realistic traffic scenarios. Subsequently, we briefly summarize the key features of CommonRoad-RL.

*1) Scenario specification:* CommonRoad-RL fully adopts the scenario specification of CommonRoad [2]. In CommonRoad, a driving scenario consists of a road network, static obstacles, dynamic obstacles, and an initial state as well as a goal region of the planning problem. In CommonRoad benchmarks, a road network is represented as *lanelets* [19], which is more lightweight and yet as expressive as other formats, such as OpenDRIVE [20]. In addition, CommonRoad provides a map converter ( [6] is included in [8]), which converts maps from OpenDRIVE [20] and OpenStreetMap [21] into lanelets and supports the manual creation of own maps in a graphical editor.

Currently, CommonRoad benchmarks offer more than 4 000 open-source scenarios, including highways, intersections, roundabouts, and race tracks. The number of scenarios is constantly increasing because the scenario factory [7] generates scenarios automatically by downloading maps from all over the world and generating traffic with the SUMO simulator [3]. In addition, we provide a tool[1] to convert popular traffic datasets, which are publicly available for scientific purposes. Currently, the supported datasets include highD [22], inD [23], and INTERACTION [24]. We will release converters for more popular datasets in the future.

*2) Configurable Markov decision process:* We model motion planning problems for autonomous vehicles as a finite-horizon, discounted MDP, defined by a 5-tuple ($\mathcal{S}$, $\mathcal{A}$, $P$, $r$, $\gamma$), where $\mathcal{S}$ represents the set of states; $\mathcal{A}$ represents the set of actions; $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ returns the transition probabilities; $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ returns the immediate reward; $\gamma \in (0, 1)$ is the discount factor. If the agent can only observe partial information from the environment, the perceived part of the state space $\mathcal{S}$ is referred to as the observation space.

Each work for RL-based motion planning for autonomous vehicles uses different state spaces, action spaces, and rewards [1]. CommonRoad-RL provides a comprehensive definition of MDPs, which are commonly used in the literature. Furthermore, users can easily modify existing MDPs using a configuration file, which allows them to easily compare different configurations. In addition, our modular architecture simplifies the creation of new MDPs from scratch if the desired ones are not provided.

*3) Core modules:* Our architecture is shown in Fig. 1 as a unified modeling language (UML) class diagram[2], where our class is inherited from the OpenAI `gym.Env` [15]. CommonRoad-RL consists of four core modules:

- `observation` contains information about the environment, including states of the ego vehicle, states of surrounding traffic participants, topological information about the road network, and information about the goal region;
- `action` defines different kinds of actions, including control inputs of different vehicle models and discrete actions for high-level planning;
- `termination` defines the conditions when an episode ends;
- `reward` provides a sparse and dense definition of reward functions.

[1] commonroad.in.tum.de
[2] uml.org

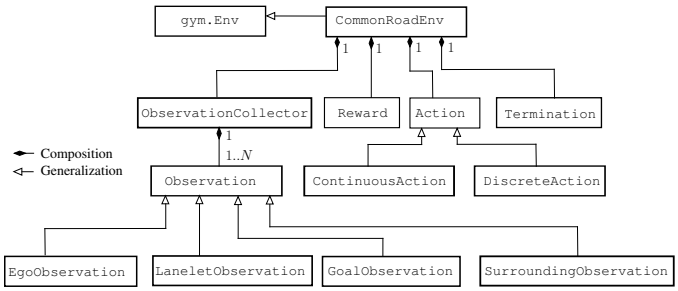



Fig. 1: UML class diagram of CommonRoad-RL.

### B. Observation Space

We define observations that are commonly used in the literature in four categories, as shown in Tab. II. Note that we represent some observations in a Frenet frame [25] aligned with the centerline of a lane occupied by the ego vehicle, whereas the global frame is a Cartesian coordinate system.

The surrounding traffic participants can either be detected by lane-based adjacency relative to the ego vehicle or by Lidar-like beams. Both detection methods are provided in CommonRoad-RL and are shown in Fig. 2. The shape and range of the sensing area as well as the number of Lidar-like beams are user-defined parameters.

### C. Action Space

The action space of MDPs can be discrete or continuous. For discrete action spaces, either continuous control inputs are discretized or high-level planning decisions are used. If high-level planning decisions are modeled by discrete spaces, a low-level controller is required to execute high-level actions. CommonRoad-RL currently features a discrete action space and multiple continuous action spaces.

Our continuous action spaces are defined by control inputs for the following vehicle models: point-mass model, kinematic single-track model, single-track model, multi-body model, and yaw-rate model. The yaw-rate model is similar to the kinematic single-track model but uses yaw-rate as input [26]. The definition of the other models can be found in the CommonRoad documentation[3].

The discrete action space of CommonRoad-RL contains longitudinal and lateral jerk inputs to a vehicle model. We discretize the jerk values as

$$j_i = -j_{\max} + \frac{2i}{n_k - 1} j_{\max},$$

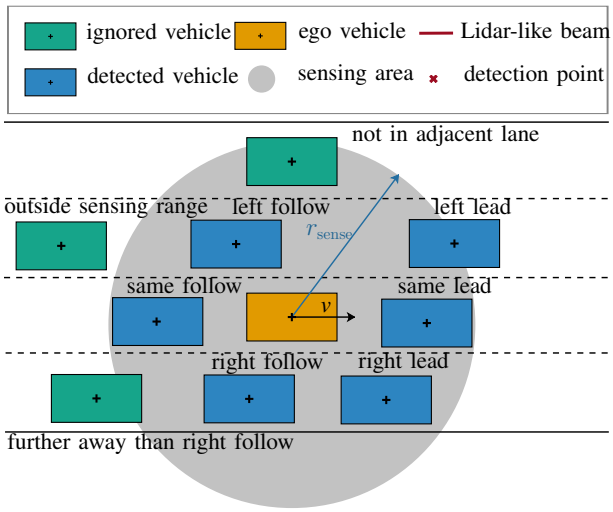
where $j_{\max}$ is the maximum jerk possible for the ego vehicle, $n_k$ represents the number of discrete actions, and $i \in \{0, \dots, n_k\}$. This is performed analogously for the lateral and longitudinal jerks. When a jerk value would result in an acceleration outside the friction circle, we project it on the friction circle to consider the friction limit of tires.

[3] commonroad.in.tum.de/model_cost_functions

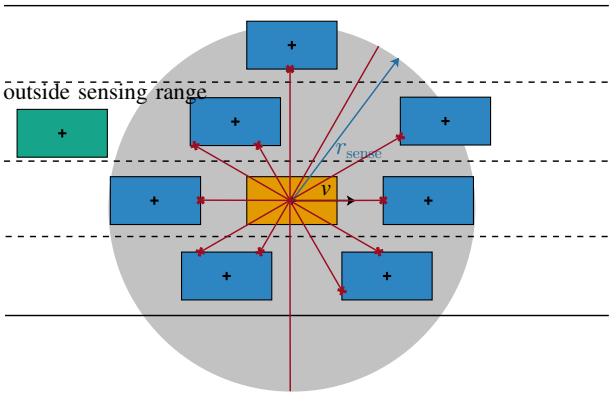
TABLE II: Observation space definition.

| Observation | Unit | Description |
|---|---|---|
| **Ego-Vehicle-Related** | | |
| velocity | $\mathrm{m/s}$ | velocity |
| acceleration | $\mathrm{m/s^2}$ | acceleration |
| jerk | $\mathrm{m/s^3}$ | jerk |
| relative heading | $\mathrm{rad}$ | yaw angle in the Frenet frame |
| steering angle | $\mathrm{rad}$ | steering angle |
| steering angular velocity | $\mathrm{rad/s}$ | steering angular velocity |
| turn rate | $\mathrm{rad/s}$ | first derivative of yaw angle in the global frame |
| **Road-Network-Related** | | |
| relative offset | $\mathrm{m}$ | offset to lane centerline; positive means right of centerline |
| lane curvature | $\mathrm{m^{-1}}$ | curvature of closest centerline point |
| left marker distance | $\mathrm{m}$ | lateral distance to left lane marking |
| right marker distance | $\mathrm{m}$ | lateral distance to right lane marking |
| left road edge distance | $\mathrm{m}$ | lateral distance to left road edge |
| right road edge distance | $\mathrm{m}$ | lateral distance to right road edge |
| off-road* | – | if vehicle drives out of roadway |
| **Goal-Related** | | |
| Euclidean goal distance | $\mathrm{m}$ | Euclidean distance to goal region |
| longitudinal goal distance | $\mathrm{m}$ | longitudinal distance to goal region in Frenet frame |
| lateral goal distance | $\mathrm{m}$ | lateral distance to goal region in Frenet frame |
| remaining time steps | – | remaining time steps defined in the goal specification |
| relative goal orientation | $\mathrm{rad}$ | smallest orientation deviation to interval of goal orientations |
| relative goal velocity | $\mathrm{m/s}$ | smallest velocity deviation to interval of goal velocities |
| goal reached* | – | if ego vehicle reaches the goal region |
| time out* | – | if there is no remaining time |
| **Surrounding-Traffic-Participants-Related** | | |
| collision* | – | if ego vehicle collides with other traffic participants |
| **Lane-based surrounding detection (see Fig. 2a)** | | |
| distance to surrounding participants | $\mathrm{m}$ | distance between ego vehicle and six lane-based surrounding traffic participants (within $r_{\mathrm{sense}}$) |
| relative velocity to surrounding participants | $\mathrm{m/s}$ | relative velocity between ego vehicle and six lane-based surrounding traffic participants |
| **Lidar-based surrounding detection (see Fig. 2b)** | | |
| Lidar-like beams 1-$N$ | $\mathrm{m}$ | distance of $N$ Lidar-like beams (within $r_{\mathrm{sense}}$) |
| range rate Lidar-like beams | $\mathrm{m/s}$ | relative velocity of reflection points |

* Boolean variables




(a) Lane-based detection for six adjacent vehicles within a circular sensing range




(b) Lidar-based surrounding detection using 12 Lidar-like beams within a circular sensing range

Fig. 2: Environment detection in CommonRoad-RL.

### D. Termination and Rewards

Users can select several termination conditions in CommonRoad-RL using the binary variables in Tab. II:

- $\mathbf{1}_{\text{reach\_goal}} = 1$ if the ego vehicle reaches the goal area.
- $\mathbf{1}_{\text{collision}} = 1$ if the ego vehicle collides with others.
- $\mathbf{1}_{\text{off\_road}} = 1$ if the ego vehicle drives off-road.
- $\mathbf{1}_{\text{time\_out}} = 1$ if the time limit of the scenario is reached.
- $\mathbf{1}_{\text{safe\_dist}} = 1$ if the safe distance between the ego vehicle and its leading vehicle is violated.

Let $c_\square$ denote coefficients for each reward term. We define a sparse reward function as:

$$r_{\text{sparse}} = r_{\text{reach\_goal}} + r_{\text{collision}} + r_{\text{off\_road}} + r_{\text{time\_out}}, \quad (1)$$

where

$$r_\square = c_\square \cdot \mathbf{1}_\square.$$

In addition, we define a dense reward function as:

$$r_{\text{dense}} = r_{\text{sparse}} + r_{\text{closer}} + r_{\text{safe\_dist}} + r_{\text{road\_center}}, \quad (2)$$

TABLE III: Related tools and their supported datasets.

| Tool | Supported datasets |
|---|---|
| **BARK** [13] | INTERACTION |
| **CRTS** [17] | NGSIM , openDD [29] |
| **CommonRoad-RL** | NGSIM, highD, inD, INTERACTION |

where

$$\begin{aligned} r_{\text{closer}} &= c_{\text{closer\_long}} \left[d_{\text{long}}(k-1) - d_{\text{long}}(k)\right] \\ &\quad + c_{\text{closer\_lat}} \left[d_{\text{lat}}(k-1) - d_{\text{lat}}(k)\right], \\ r_{\text{safe\_dist}} &= -\exp(c_{\text{safe}} \frac{d_{\text{lead}}}{d_{\text{safe}}})\, \mathbf{1}_{\text{safe\_dist}}, \\ r_{\text{road\_center}} &= c_{\text{center}} |d_{\text{lat\_offset}}|, \end{aligned}$$

where $d_{\text{long}}(k)$ and $d_{\text{lat}}(k)$ denote the longitudinal and lateral distance between the ego vehicle and the goal region at time $k \in \mathbb{N}$, respectively. Furthermore, $d_{\text{lead}}$ and $d_{\text{safe}}$ denote the current distance and safe distance [27] between the ego vehicle and its leading vehicle, respectively, and $d_{\text{lat\_offset}}$ denotes the lateral offset of the ego vehicle to the center line of its current lane. Note that we use an exponential function in $r_{\text{safe\_dist}}$ to severely penalize agents driving too close to the leading vehicle.

## IV. Experiments

We demonstrate the usefulness of CommonRoad-RL by comparing four agents trained with different settings[4]. The relevant configuration parameters are listed in the appendix. We use the deep RL algorithm *proximal policy approximation* (PPO) [28]. All experiments are conducted on the real-world highway drone (highD) dataset [22], which contains 16.5 h of vehicle trajectories with a time step size of $\Delta t$ = 0.04 s. We convert the dataset into 3000 scenarios with a duration of 40 s. For each scenario, we randomly choose a vehicle as the ego vehicle, create a planning problem using its initial and final states, and remove this vehicle from the scenario.

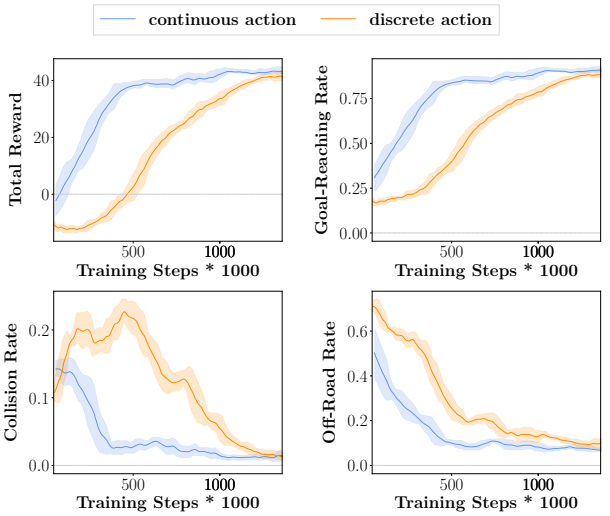
Note that we omit an experimental comparison between our tool and other tools because none of the existing tools supports the same settings we used in our experiments. For interested readers, we list the necessary adaptions of existing tools after each experiment. We limit ourselves to tools that support real-world scenarios and provide an RL training environment, namely, BARK [13] and CARLA Real Traffic Scenarios [17]. The supported datasets are listed in Tab. III.

### A. Reward Comparison

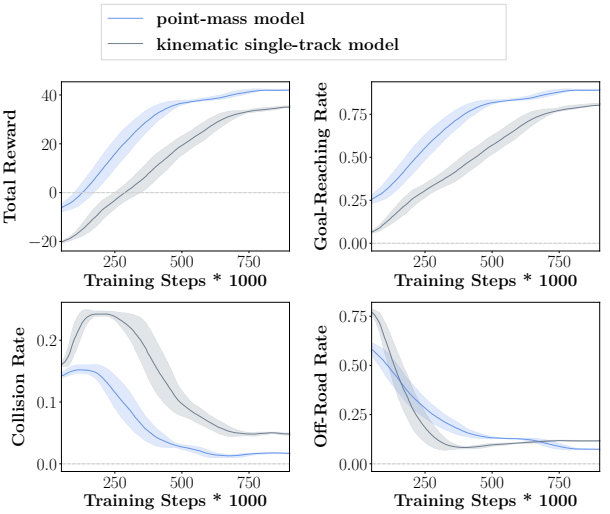
The learning curves of agents trained with the sparse reward (1) (named *sparse agent*) and the dense reward (2) (named *dense agent*) are shown in Fig. 3a. Note that both agents are trained using a continuous action space and a point-mass vehicle model. We omit the learning curves of

[4]All configurations used in our experiments are provided in gitlab.lrz.de/tum-cps/commonroad-rl to reproduce our results.


(a) Learning curves of agents trained with sparse and dense rewards.




(b) Learning curves of agents trained with discrete and continuous actions.




(c) Learning curves of agents trained with point-mass model and kinematic single-track model.

Fig. 3: Learning curves of agents trained with different reward functions, action spaces, and vehicle models.

reward values as these two reward definitions have different scales. The goal-reaching rate of the sparse agent is significantly higher than the dense agent, and the collision and off-road rates of the sparse agent are significantly lower than for the dense agent. Although the time-out rate of the sparse agent is slightly higher than that of the dense agent, it can be concluded that the sparse agent significantly outperformed the dense agent. A possible reason is that the dense reward is more sensitive to the coefficients of each term in (2). Therefore, using the default coefficients was insufficient for the agent to converge to good behavior. To improve the performance of the dense agent, an automatic parameter search can be performed for the reward coefficients using CommonRoad-RL.

BARK supports a similar sparse and dense reward, which can be selected by changing the source code of the configurable environment. In addition, the coefficients of the reward terms are easily adaptable through a configuration file. On the other hand, CARLA Real Traffic Scenarios also support both reward types even though their coefficients cannot be configured directly.

### *B. Action Comparison*

To compare different action spaces, we used the continuous action space of the point-mass model and the discrete action space described in Sec. III-C, where $n_i = n_{\text{long}} = n_{\text{lat}} = 3$ and $j_{\max} = 10\,\text{m}\,\text{s}^{-3}$. This results in nine possible action combinations. The agent can select a new discrete action for every discrete time step. Fig. 3b shows learning curves for both action spaces when training with the sparse reward. Using a discrete action space exhibits a slower convergence than using a continuous action space. However, the total reward, goal-reaching rate, collision rate, and off-road rate of both agents are similar after $1.5\,\text{million}$ training steps. The slower convergence is most likely due to the discrete agent's need to learn an optimal behavior with fewer available actions.

In CARLA Real Traffic Scenarios, the action space is a `carla.Vehicle` object, which requires an additional interface to convert the raw action values into the required object. Furthermore, this interface needs to be integrated with the source code of the learning algorithms, which reduces modularity. To compare discrete and continuous action spaces, both action spaces need to be implemented in this interface. Although BARK contains both types of action spaces, the exact implementation differs and the source code of the configurable environment needs to be adapted to switch between the discrete and continuous action spaces.

### *C. Vehicle Model Comparison*

The learning curves of two agents trained using a point-mass model and a kinematic single-track model, respectively, are shown in Fig. 3c. Note that both agents are trained using the sparse reward and a continuous action space. The agent trained using a point-mass model performed moderately better in terms of reward values, goal-reaching rates, and collision rates. The off-road rates of the two agents are almost

identical. The different performance of the two agents could result from the fact that the point-mass model neglects the nonholonomic behavior of vehicles, and thus, it can reach the goal and avoid collisions more easily. However, some of the solutions of the point-mass model might not be drivable because nonholonomic behavior is not considered.

The point-mass model and kinematic single-track model are generally available in BARK, yet in the RL interface of BARK, only the kinematic single-track is used for continuous planning. Additional models would have to be implemented, and switching between different vehicle models would be more difficult than in CommonRoad-RL. CARLA provides different vehicle models, yet it still lacks the point-mass and kinematic single-track models.

### D. Discussion

One million training steps take approximately 2000 s on 32 threads on a machine with an AMD EPYC 7742 2.2 GHz processor and 1024 GB of DDR4 3200 MHz memory. This computation time is viable for testing different configurations. Our experiments show that the agents trained on the highD dataset achieved a 40% to 85% goal-reaching rate using the default setting of the observation space and reward coefficients. In addition, the results show that the reward definition has a larger effect on performance than the action space definition and used vehicle models.

Our modular implementation makes it easier to improve these results as it enables automatic fine-tuning of configuration parameters for the learning algorithm and environment. In addition, each experiment was conducted by simply changing one parameter in the configuration file, whereas 60 parameters can be adapted. Thus, many different settings can be examined with CommonRoad-RL, and benchmarking different MDPs for motion planning in autonomous driving is straightforward. On the contrary, in BARK and CARLA Real Traffic Scenarios, the observation spaces are much less expressive, and implementing additional missing features is required.

## V. Conclusions

We introduced CommonRoad-RL—an open-source toolbox to train and evaluate RL-based motion planners for autonomous vehicles. The configurable state spaces, action spaces, and rewards of CommonRoad-RL enable researchers to create and identify an ideal choice of an MDP for their autonomous driving task automatically. To illustrate this, we compared agents trained with different reward definitions, vehicle models, and action definitions.

To provide additional benefits, we will provide the following functionalities in CommonRoad-RL in the near future:

- multi-agent RL for cooperative driving with an interface to allow individual actions for each agent;
- closed-loop simulation: we will add simulation to the existing interface with the traffic simulator SUMO [4], an interface to CARLA, and integrate behavior models directly into CommonRoad-RL;
- support for more scenario specifications, such as Lanelet2 [30] and OpenSCENARIO[5].
- open-source wrappers for safety layers [31] to ensure safety during learning;
- uncertainties in observations.


## Acknowledgment

The authors gratefully acknowledge the partial financial support of this work by the German Research Foundation Grant AL 1185/3-2 and the research training group CONVEY funded by the German Research Foundation under grant GRK 2428.


## References


[1] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–20, 2020.

[2] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719 – 726.

[3] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using SUMO," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2575–2582.

[4] M. Klischat, O. Dragoi, M. Eissa, and M. Althoff, "Coupling sumo with a motion planning framework for automated vehicles," in *SUMO User Conference*, 2019.

[5] C. Pek, V. Rusinov, S. Manzinger, M. C. Üste, and M. Althoff, "CommonRoad drivability checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 1013–1020.

[6] M. Althoff, S. Urban, and M. Koschi, "Automatic conversion of road networks from OpenDRIVE to lanelets," in *Proc. of the IEEE International Conference on Service Operations and Logistics, and Informatics*, 2018, pp. 157–162.

[7] M. Klischat, E. Irani Liu, F. Höltke, and M. Althoff, "Scenario factory: Creating safety-critical traffic scenarios for automated vehicles," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2020.

[8] S. Maierhofer, M. Klischat, and M. Althoff, "CommonRoad Scenario Designer: An Open-Source Toolbox for Map Conversion and Scenario Creation for Autonomous Vehicles," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2021.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. of the 1st Annual Conference on Robot Learning*.

[10] "LGSVL Simulator: An Autonomous Vehicle Simulator," https://github.com/lgsvl/simulator, LG Electronics America R & D Center, 2020, (visited on Mar. 11, 2021).

[11] "Automotive Simulator," https://github.com/sisl/AutomotiveSimulator.jl, Stanford Intelligent Systems Laboratory, 2020, (visited on Mar. 11, 2021).

[12] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS, The Open Racing Car Simulator," http://www.torcs.org, 2014, (visited on Mar. 11, 2021).

[13] J. Bernhard, K. Esterle, P. Hart, and T. Kessler, "BARK: Open Behavior Benchmarking in Multi-Agent Environments," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 6201–6208.

[14] E. Leurent, "An Environment for Autonomous Driving Decision-Making," https://github.com/eleurent/highway-env, 2018, (visited on Mar. 11, 2021).

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[16] K. Tong, Z. Ajanovic, and G. Stettinger, "Overview of Tools Supporting Planning for Automated Driving," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2020.


[5] openscenario.org

[17] B. Osiński, P. Miłoś, A. Jakubowski, P. Zięcina, M. Martyniak, C. Galias, A. Breuer, S. Homoceanu, and H. Michalewski, "CARLA Real Traffic Scenarios – novel training ground and benchmark for autonomous driving," *arXiv preprint arXiv:2012.11329*, 2020.
[18] "Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data," data.transportation.gov, U.S. Department of Transportation Federal Highway Administration, 2016, (visited on Mar. 11, 2021).
[19] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.
[20] M. Dupuis, M. Strobl, and H. Grezlikowski, "OpenDRIVE 2010 and beyond – status and future of the de facto standard for the description of road networks," in *Proc. of the Driving Simulation Conference Europe*, 2010, pp. 231–242.
[21] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.
[22] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2118–2125.
[23] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The inD dataset: A drone dataset of naturalistic road user trajectories at German intersections," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 1929–1934.
[24] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," *arXiv:1910.03088 [cs, eess]*, 2019.
[25] M. P. Do Carmo, *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.
[26] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*, J. P. Laumond, Ed. Springer Berlin Heidelberg, 1998, pp. 171–253.
[27] M. Althoff and R. Lösch, "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" in *Prof. of the IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 485–491.
[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[29] A. Breuer, J.-A. Termöhlen, S. Homoceanu, and T. Fingscheidt, "openDD: A large-scale roundabout drone dataset," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2020.
[30] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 1672–1679.
[31] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2020.

## APPENDIX

### CONFIGURATIONS USED IN THE NUMERICAL EXPERIMENTS

```
# vehicle model and types
vehicle_params:
  #1: FORD_ESCORT; 2: BMW_320i; 3: VW_VANAGON
  vehicle_type: 2
  #0: PM, 1: ST, 2: KS, 3: MB, 4: YawRate
  vehicle_model: 0

# action configuration
action_configs:
  action_type: discrete # continuous
  long_steps: 3
  lat_steps: 3

# reward configuration
reward_type: sparse_reward
reward_configs:
  reward_goal_reached: 50.
  reward_collision: -50.
  reward_off_road: -20.
  reward_time_out: -10.
  # only used in dense_reward
  reward_closer_to_goal_coefficient_long: 0.5
  reward_closer_to_goal_coefficient_lat: 0.5
  reward_safe_distance_coefficient: 5.
  reward_stay_in_road_center: 1.
```