Safe Reinforcement Learning for Automated Vehicles via Online Reachability Analysis

Xiao Wang and Matthias Althoff, *Member, IEEE*

***Abstract*—Ensuring safe and capable motion planning is paramount for automated vehicles. Traditional methods are limited in their ability to handle complex and unpredictable traffic situations. Model-free reinforcement learning (RL) addresses this challenge by generalizing across different traffic situations without requiring explicit knowledge of all possible outcomes. However, it also poses challenges due to its inherent lack of safety guarantees. To bridge this gap, we integrate online reachability analysis into model-free RL to provide real-time safety guarantees. Reachability analysis helps to identify unsafe states and actions, enabling provably safe decision-making in automated vehicles. We evaluate the effectiveness of our approach through extensive numerical experiments. Our results demonstrate that we can efficiently provide safety guarantees without impairing the performance of the learned agent.**

***Index Terms*—Automated vehicles, motion planning, reinforcement learning, reachability analysis, safety verification, and formal methods.**

I. Introduction

AUTOMATED vehicles provide a promising solution to improve safety and traffic efficiency while reducing energy consumption [1]. Various motion planning techniques have been developed for automated vehicles in the last decades [2]. Among these approaches, data-driven methods have shown great potential to handle complex traffic scenarios with much less required expert knowledge compared to rule-based methods [3], [4]. Motion planning can be modeled as sequential decision-making problems, for which reinforcement learning (RL) techniques have achieved enormous advances in recent years [5]–[7]. However, due to its trial-and-error nature, RL methods lack safety guarantees, which prevents them from being applied to real-world safety-critical systems, such as automated vehicles.

Safe RL has thus become a very active research area aiming to improve or guarantee the safety of RL-based controllers and motion planners during training and/or deployment. Gu et al. have reviewed recent advances in safe RL methods for motion planners of automated vehicles in [8, Sec. 5.1]. However, to the best knowledge of the authors, none of the existing works offers a hard safety guarantee for RL-based planners with continuous action spaces for an infinite time horizon, nor do they compare safety verification techniques for safe RL. In this article, we propose a safe RL framework based on online reachability analysis [9], [10] and compare the proposed method with another safe RL approach using control barrier functions [11]. The proposed framework is shown in Fig. 1 and explained in more detail in Sec. III. We demonstrate through our numerical experiments that reachability-based methods are more suitable for longer planning horizons compared to approaches using control barrier functions. In addition, control barrier functions are designed for specific cases, e.g., highway scenarios [11], while reachability analysis works for arbitrary traffic situations [9].

This work is funded by the European Commission project justITSELF under grant number 817629. Xiao Wang and Matthias Althoff are with the School of Computation, Information and Technology, Technical University of Munich, D-85748 Garching, Germany (e-mail: {xiao.wang, althoff}@tum.de).

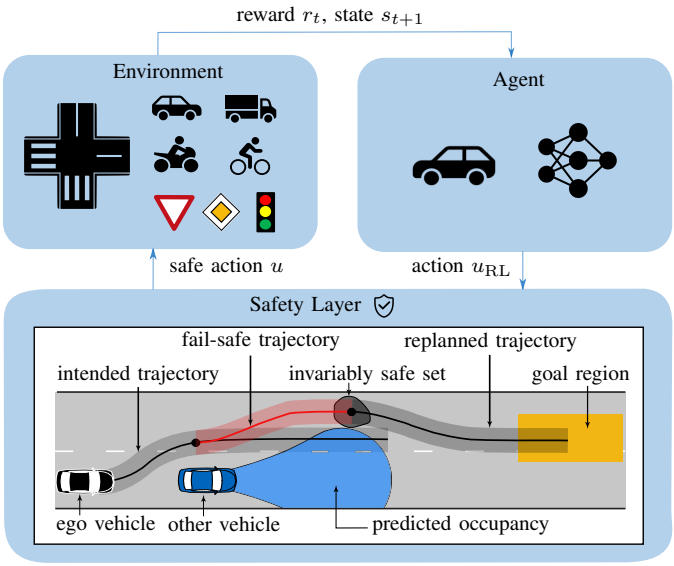


Fig. 1: Overview of the proposed approach. The RL agent returns an action that generates an intended trajectory. Based on reachability analysis, the safety layer predicts the occupancies of other traffic participants and changes the intended trajectory to the closest safe trajectory. The RL agent receives a reward, replans from a new safe state, and eventually reaches a desired goal area.

A. Literature Review

Subsequently, we summarize related surveys on safe RL methods and their applications to autonomous driving. A detailed review of safety verification methods for automated vehicles is provided in [12, Sec. 3] and thus omitted in this article. In conclusion of [12], reachability analysis is the most common and popular method for online verification for arbitrary traffic situations and is thus utilized in this article.

a) Surveys on Safe RL: In recent years, many surveys on safe RL algorithms emerged. For instance, García et al. [13] categorize safe RL approaches into two groups: those that modify the optimality criterion and those that modify the exploration process. Similarly, Brunke et al. [14] review safe learning control methods, examining the level of safety guarantees provided as well as the reliance on prior knowledge

TABLE I: Literature on safe RL for automated vehicles organized by action space and used safety methods. Abbreviations used here are listed in Table II.

Action Space	Safety/ Shielding Method	RL Method	Applied Scenario	Type of Guarantee	Explicit Prediction of Other Traffic Participants	Ref.
discrete	action masking	DQN	highway	guarantee for an infinite time	based on physical limits and traffic rules	[17]
		Q-learning	ACC	guarantee when model is accurate	no	[18]
		Double DQN	highway	no formal proof of safety	no	[19]
		DQN	intersection	guarantee with a certain probability	most likely behavior	[20]
		DQN	intersection	guarantee with a certain probability	no	[21]
		DQN	intersection	guarantee with a certain probability	no	[22]
		PPO	highway	guarantee for an infinite time	set-based reachability analysis	[23]
		PPO	highway, roundabout, intersection	formally proved safety for linearized system dynamics	no	[24]
		DQN	intersection	no: prediction might be inaccurate	a collision predictor based on LSTM	[25]
		Inverse RL	highway	guarantee for an infinite time	based on physical limits and traffic rules	[26]
		PPO	intersection	guarantee for an infinite time	set-based reachability analysis	[27]
	action replacement	IQN	intersection	no: constant safety gap is not formally correct	worst-case assumption	[28]
		Double DQN	highway	no: considered safe distance assumes constant velocity	constant turn rate and acceleration	[29]
		Double DQN	highway	no: prediction might be inaccurate	regret theory to predict lane changes	[30]
		DQN	intersection	no: assumes braking is always safe for intersections	RSS	[31]
	constrained RL	parallel CPO	intersection	guarantee for the considered horizon	N/A: centralized policy for all vehicles	[32]
		Q learning	highway	guarantee for the considered horizon	N/A: consider static obstacles only	[33]
	reward shaping	policy gradient	double merge	no[1]	N/A	[34]
		PPO	urban	no[1]	a simple prediction model assuming accelerations follow a uniform distribution	[35]
		MCTS	highway	no[1]	no	[36]
		DQN	merging	no guarantee for maneuvers that use reward shaping[1]	worst-case assumption	[37]
	risk-aware RL	QRDQN	intersection	no[1]	no	[38]
		DQN	intersection	no[1]	worst-case assumption	[39]
continuous	action projection	DDPG	racing	no: generate artificial potential field based on current states, no prediction	no	[40]
		DDPG	ACC	no: constant distance and time headway	no	[41]
		PPO	highway	guarantee for one time step	worst-case assumption	[11]
		PPO	highway	guarantee for the considered horizon	set-based reachability analysis	[42]
	action replacement	TD3	highway	guarantee for the considered horizon	N/A: consider static obstacles only	[43]
		attention PPO	highway	guarantee for one time step	reachable velocity set given acceleration and steering angle bounds	[44]

[1] By nature, reward shaping and risk-aware RL can only improve safety instead of guaranteeing safety.

TABLE II: Abbreviation used in this article.

Abbr.	Meaning	Ref.
ACC	Adaptive Cruise Control	N/A
CPO	Constrained Policy Optimization	[45]
DDPG	Deep Deterministic Policy Gradient	[46]
DQN	Deep Q-Network	[5]
IQN	Implicit Quantile Networks	[47]
LSTM	Long Short-Term Memory	[48]
MCTS	Monte Carlo Tree Search	N/A
MLP	Multi-Layer Perceptron	[49]
PPO	Proximal Policy Optimization	[50]
QRDQN	Quantile Regression DQN	[51]
RSS	Responsibility-Sensitive Safety	[52]
TD3	Twin Delayed DDPG	[53]

and data. Krasowski et al. [15] focus on provably safe RL methods that use an external layer to mask or correct potential unsafe actions from an RL agent. Gu et al. [8] provide a theoretical and practical overview of safe RL techniques, encompassing model-based and model-free approaches. Lastly, Zhao et al. [16] distinguish state-wise safe RL algorithms, where the problem is modeled as a state-wise constrained Markov decision process, based on whether safety is ensured after convergence or during training. State-wise RL focuses on individual states instead of the average rewards or costs over all states.

b) Safe RL for Autonomous Driving: Based on the previously mentioned surveys, we categorize literature on safe RL for automated vehicles in Table I according to the adopted safety/shielding methods and the considered action space. Furthermore, we comprehensively list the utilized RL method, the specific traffic scenario in which it was applied, the types of safety guarantees obtained, and whether an explicit prediction of other traffic participants was incorporated. From Table I, we can conclude that most works focus on discrete and finite

action spaces. However, in such cases, a low-level trajectory planner is often required, and the RL agent might converge to a suboptimal performance due to discretization effects. Approaches for continuous action spaces are comparatively less studied. In addition, methods that modify the optimization objective, such as reward shaping, constrained RL, and risk-aware RL, lack robust safety guarantees when dynamic obstacles are present, because safety is only considered a soft constraint in such cases. Lastly, safety guarantees of methods that rely on external safety shields [15], i.e., action masking, action replacement, and action projection, highly depend on the predictions of other traffic participants. Many works assume fixed behavior patterns for simplicity [29] or predict the most likely behavior [20], [30]. Obviously, the actual behavior might substantially deviate from particular predictions. Consequently, safety cannot be guaranteed in such cases. Alternatively, some methods assume worst-case behaviors within a lane, which are more suitable for short time horizons but are impractical for longer time horizons, as later shown in Fig. 9a. In addition, worst-case assumptions only work for monotonic systems and do not work for the non-monotone lateral dynamics of a vehicle, as shown in [10, Fig. 10]. On the contrary, set-based predictions using reachability analysis [54] provide an over-approximation of all feasible future movements of other traffic participants that adhere to traffic rules; thus, safety can be guaranteed.

Among all previous works, the closest to ours are [11], [42]. Kochdumper et al. [42] propose a reachability-based safety shield for RL controllers for general nonlinear systems. However, for the case study of automated vehicles, [42] only ensures collision avoidance during the planning horizon. Furthermore, the set representations used in [42] are suitable for general systems but are less efficient for our application; thus, the safety shield is only applied during deployment for the autonomous driving use case. Compared to [42], our approach performs online reachability analysis and thus can be applied during training as well. Wang [11] proposed a safety shield using control barrier functions to ensure safety. However, the control barrier functions proposed in [11] only focus on highway scenarios and rely on the worst-case assumption of other traffic participants, which is more suitable for short planning horizons, as we show empirically in Sec. IV.

B. Contributions

We propose a method to safeguard RL-based motion planners with a continuous action space. Compared to our previous methods [11], [42], this approach works for arbitrary planning horizons, does not require offline reachable set computation, and can guarantee safety for infinite time. To demonstrate the effectiveness of our method, we perform an extensive numerical evaluation using real-world datasets. Furthermore, we provide the first comparison of reachability analysis with control barrier functions for safe reinforcement learning.

C. Outline

The remainder of this paper is structured as follows: In Sec. II, we introduce the notation used, the vehicle dynamics, basic concepts in RL, and our problem statement. We present our solution concept based on the online verification framework in Sec. III, which is evaluated using real-world highway scenarios in Sec. IV. Finally, we draw conclusions in Sec. V.

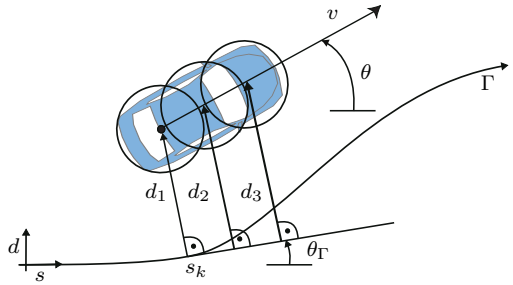


Fig. 2: Kinematic single-track model in a curvilinear coordinate system, where Γ corresponds to a given drivable reference path, e.g., the centerline of a lane occupied by the ego vehicle. The reference point of the vehicle is the center of the rear axle.

II. Preliminaries

A. Notation

This work uses discrete time $t_k = k\Delta t$ with a constant time step size $\Delta t \in \mathbb{R}_+$, where k is the time step. We use $\square_k$ to denote the value of variable $\square$ at time t_k. Note that $\square$ serves as a placeholder for variables in this article. Similarly, $\square_{[k,k+f]}$ denotes the discrete-time trajectory of a variable for the time interval $[t_k, t_{k+f}]$, i.e., $\square_{[k,k+f]} = [\square_k, \square_{k+1}, \cdots, \square_{k+f}]$. Let $x \in \mathcal{X} \subset \mathbb{R}^n$ denote the set of admissible states and $u \in \mathcal{U} \subset \mathbb{R}^m$ the set of admissible inputs. We consider a dynamical system of the form

$$x_{k+1} = f(x_k, u_k). \tag{1}$$

The notation $x_k(x_0, u_{[0,k]})$ represents the state of system (1) at time t_k when starting from an initial state x_0 subject to the input trajectory $u_{[0,k]}$. The superscript $x^{(i)}, i \in \mathbb{N}_+$ describes the i-th entry of the state variable x, whereas the subscripts $\square_{\mathrm{lon}}$ and $\square_{\mathrm{lat}}$ respectively describe a variable in the longitudinal and lateral direction of a curvilinear coordinate system for a given reference path Γ illustrated in Fig. 2. We assume that Γ is always drivable. Note that we denote the control input as u and the actions of the RL agent as a_{RL}, which encodes trajectories of control inputs in our setting, as detailed later in (22). We use the projection operator $\mathrm{proj}_{\square}(x)$ to map a state $x \in \mathcal{X}$ to its elements specified by $\square$. The projection of a set is then represented as $\mathrm{proj}_{\square}(\mathcal{X}) := \{\mathrm{proj}_{\square}(x) | \, x \in \mathcal{X}\}$. The operator $\mathrm{occ}(x) : \mathbb{R}^n \to \mathbb{R}^2$ relates the state x of the ego vehicle to its occupancy. In addition, we define the occupancy $\mathcal{O}_k$ of all relevant traffic participants and the one-step reachable set $\mathcal{R}_{k+1}$ as well as the drivable area $\mathcal{D}_k$ of the ego vehicle:

Definition 1 (Occupancy $\mathcal{O}_k$)
The occupancy $\mathcal{O}_{i,k} \subseteq \mathbb{R}^2$ represents the set of positions that an obstacle i could potentially occupy at time step k. Let $\mathcal{B} \subset \mathbb{N}$ denote the set of all relevant traffic participants in the considered traffic scenario. We define $\mathcal{O}_k = \bigcup_{i \in \mathcal{B}} \mathcal{O}_{i,k}$.

Note that we model the road boundaries as static obstacles in a traffic scenario, see Fig. 3b. Therefore, avoiding collision with $\mathcal{O}_k$ also ensures driving within the road boundaries.

Definition 2 (One-Step Reachable Set $\mathcal{R}_{k+1}$)
The reachable set $\mathcal{R}_{k+1}$ is the set of states reachable from $\mathcal{R}_k$ within one step without its occupancy intersecting the occupancy of obstacles $\mathcal{O}_{k+1}$:

$$\mathcal{R}_{k+1} := \big\{x_{k+1} \in \mathcal{X} \,\big|\, \exists\, x_k \in \mathcal{R}_k,\, \exists\, u_k \in \mathcal{U} : \\ x_{k+1} = f(x_k, u_k), \mathrm{occ}(x_{k+1}) \cap \mathcal{O}_{k+1} = \emptyset\big\}.$$

Definition 3 (Drivable Area $\mathcal{D}_k$)
The drivable area $\mathcal{D}_k$ of the ego vehicle is the projection of its reachable set $\mathcal{R}_k$ onto the position domain: $\mathcal{D}_k := \mathrm{proj}_{s,d}(\mathcal{R}_k)$, where s and d denote the longitudinal and lateral positions, respectively.

Before introducing invariably safe sets, we define the set of collision-free states $\mathcal{F}_k$ as:

Definition 4 (Collision-free Set $\mathcal{F}_k$)
The set of collision-free states $\mathcal{F}_k \subseteq \mathcal{X}$ describes the largest possible set of states of (1) *whose associated occupancy does not collide with the occupancy of any obstacles at time step k: $\mathrm{occ}\big(\mathcal{F}_k\big) \cap \mathcal{O}_k = \emptyset$.*

When only ensuring collision avoidance during the planning horizon, the ego vehicle might end up in a situation where a collision is inevitable, e.g., when the ego vehicle is too close to a static obstacle with a high velocity. To circumvent this issue, we utilize invariably safe sets proposed in [55], which we shortly recap: A state x_k is invariably safe if a collision-free input trajectory $u_{[k,k+\tau]}$ exists which leads to another invariably safe state x_τ. We use the shorthand $u_{[k,\tau]} \in \mathcal{U}$ for $\forall i \in [k,\tau] : u_i \in \mathcal{U}$. The invariably safe set $\mathcal{S}_k$ is formally defined as follows:

Definition 5 (Invariably Safe Set $\mathcal{S}_k$)
The invariably safe set $\mathcal{S}_k$ at time step k is the maximal set of states, which allows (1) *to be safe for an infinite time horizon:*

$$\mathcal{S}_k := \big\{x_k \in \mathcal{F}_k \,\big|\, \forall \tau > k,\, \exists u_{[k,\tau]} \in \mathcal{U} : x_\tau\big(x_k, u_{[k,\tau]}\big) \in \mathcal{F}_\tau\big\}.$$

We use polytopes to under-approximate invariably safe sets.

Definition 6 (Polytope)
Given a matrix of normal vectors $A \in \mathbb{R}^{p\times n}$ and the offset $b \in \mathbb{R}^p$, the halfspace representation of a polytope $\mathcal{P} \subseteq \mathbb{R}^n$ is $\mathcal{P} := \big\{x \in \mathbb{R}^n \;\big|\; A\,x \leq b\big\}$. We use the notation $\mathcal{P} = \langle A, b\rangle_\mathcal{P}$.

B. Vehicle Dynamics in a Curvilinear Coordinate System

To capture the non-holonomic behavior of a vehicle, we choose the widely-used kinematic single-track model for planning input trajectories, which combines the front and rear axle wheels of a vehicle into a single wheel at the center of each axle. We linearize the vehicle dynamics by separating the longitudinal and lateral dynamics of the vehicle, as proposed in [56, Sec. V], which enables us to formulate a convex optimization problem. We formulate the vehicle dynamics in a curvilinear coordinate system, illustrated in Fig. 2. Note that this model only provides kinematic constraints when optimizing an input trajectory, while the safety constraints are generated later using an abstract model. Therefore, our safety concept also works with other vehicle models.

1) Longitudinal Dynamics: The longitudinal state of the vehicle is $x_\mathrm{lon} = [s, v, a, j]^T$, where s is the longitudinal position, v is the velocity, a is the acceleration, and j is the jerk of the vehicle. To obtain comfortable longitudinal motions, we use $u_\mathrm{lon} = \dddot{a}$ as the input and thus obtain

$$x_{\mathrm{lon},k+1} = \underbrace{\begin{bmatrix} 1 & \Delta\mathrm{t} & \frac{1}{2}\Delta\mathrm{t}^2 & \frac{1}{6}\Delta\mathrm{t}^3 \\ 0 & 1 & \Delta\mathrm{t} & \frac{1}{2}\Delta\mathrm{t}^2 \\ 0 & 0 & 1 & \Delta\mathrm{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A_\mathrm{lon}} x_{\mathrm{lon},k} + \underbrace{\begin{bmatrix} \frac{1}{24}\Delta\mathrm{t}^4 \\ \frac{1}{6}\Delta\mathrm{t}^3 \\ \frac{1}{2}\Delta\mathrm{t}^2 \\ \Delta\mathrm{t} \end{bmatrix}}_{b_\mathrm{lon}} u_{\mathrm{lon},k}. \tag{2}$$

In addition, we consider the subsequent time-invariant constraints for the state and input:

$$\mathcal{X}_\mathrm{lon,ti} := \mathbb{R} \times \begin{bmatrix} v_\mathrm{min}, v_\mathrm{max} \\ a_\mathrm{min}, a_\mathrm{max} \\ j_\mathrm{min}, j_\mathrm{max} \end{bmatrix}, \tag{3}$$

$$\mathcal{U}_\mathrm{lon,ti} := \Big[\dddot{a}_\mathrm{min}, \dddot{a}_\mathrm{max}\Big]. \tag{4}$$

2) Lateral Dynamics: The lateral state of the vehicle is described by $x_\mathrm{lat} = [d, e_\theta, \kappa, \dot{\kappa}]^T$, where d is the lateral position, e_θ is the relative orientation $\theta - \theta_\Gamma$, κ is the curvature, and $\dot{\kappa}$ is the derivative of the curvature. Similarly to the longitudinal motion, we use $u_\mathrm{lat} = \ddot{\kappa}$ as the input to obtain a smooth lateral behavior:

$$x_{\mathrm{lat},k+1} = \underbrace{\begin{bmatrix} 0 & v_k & 0 & 0 \\ 0 & 0 & v_k & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \end{bmatrix}}_{A_{\mathrm{lat},k}} x_{\mathrm{lat},k} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{b_\mathrm{lat}} u_{\mathrm{lat},k}. \tag{5}$$

Note that in contrast to [56, Sec. V], we add a constraint for the ego vehicle to approximately follow the pre-defined reference path Γ, i.e., e_θ is negligibly small, which enables us to utilize the small-angle assumptions $\sin(e_\theta) \approx e_\theta$ and $\cos(e_\theta) \approx 1$. Furthermore, we take into account the physical limits of the steering system of a vehicle by including the following constraints:

$$\mathcal{X}_{\mathrm{lat,ti},k} := \mathbb{R} \times \begin{bmatrix} e_{\theta,\mathrm{min}}, e_{\theta,\mathrm{max}} \\ \kappa_{\mathrm{lim,min}}, \kappa_{\mathrm{lim,max}} \\ \dot{\kappa}_\mathrm{min}, \dot{\kappa}_\mathrm{max} \end{bmatrix}, \tag{6}$$

$$\mathcal{U}_\mathrm{lat,ti} := \Big[\ddot{\kappa}_\mathrm{min}, \ddot{\kappa}_\mathrm{max}\Big], \tag{7}$$

where the value of $e_{\theta,\mathrm{min}}$ and $e_{\theta,\mathrm{max}}$ are determined such that the small-angle assumptions have less than 1% relative error. In addition, we derive the constraints for the curvature based on the friction circle (aka Kamm's circle):

$$\kappa_{\mathrm{lim,min}} = \max\left(\kappa_\mathrm{min}, -\frac{\sqrt{a_\mathrm{max}^2 - a_k^2}}{v_k}\right), \tag{8}$$

$$\kappa_{\mathrm{lim,max}} = \min\left(\kappa_\mathrm{max}, \frac{\sqrt{a_\mathrm{max}^2 - a_k^2}}{v_k}\right).$$

Note that since we decouple the longitudinal and lateral planning, i.e., v_k and a_k are known after we plan a trajectory for the longitudinal direction, constraint (6) remains linear. We list all parameters used in this article in Table VI.

3) Abstract Model for Reachability Analysis: To reduce computational cost when computing the reachable set of the ego vehicle and other traffic participants, we use the following abstract model:

$$x_{k+1} = \begin{bmatrix} 1 & \Delta\mathrm{t} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta\mathrm{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{1}{2}\Delta\mathrm{t}^2 & 0 \\ \Delta\mathrm{t} & 0 \\ 0 & \frac{1}{2}\Delta\mathrm{t}^2 \\ 0 & \Delta\mathrm{t} \end{bmatrix} u_k, \quad (9)$$

the reachable sets of which are over-approximative; the state is $x = [s, v_{\mathrm{lon}}, d, v_{\mathrm{lat}}]^T$ and the input is $u = [a_{\mathrm{lon}}, a_{\mathrm{lat}}]^T$. To ensure that the abstract model enclose all possible behaviors of a real vehicle, conformance checking using the data recorded from the real vehicle should be performed [57].

C. RL Algorithm

Reinforcement learning is performed based on a Markov decision process (MDP), formalized by a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. Here, the state space $\mathcal{S}$ comprises continuous states s_{RL}, and the action space $\mathcal{A}$ consists of continuous actions a_{RL}. The transition function $p(s'_{\mathrm{RL}}|s_{\mathrm{RL}}, a_{\mathrm{RL}})$ represents the probability distribution of arriving at state s'_{RL} from state s_{RL} by performing action a_{RL}. In addition, r denotes the immediate reward signal, and $\gamma \in [0, 1]$ represents the discount factor.

RL focuses on training agents to make decisions through interaction with an environment. An agent learns to perform actions that maximize cumulative rewards over time based on feedback from the environment. A mapping from states to actions describing the behavior of the agents is called a *policy*, denoted by $\pi_\phi(a_{\mathrm{RL}}|s_{\mathrm{RL}})$ and parameterized by ϕ. For brevity, we just write π and π' to denote policies parameterized by ϕ and ϕ' respectively throughout this article. We focus on stochastic policies in this work, which returns probability distributions for the action given the state.

Policy gradient methods are a class of RL algorithms that optimize the policy directly by repeatedly estimating the gradient of the expected reward with respect to the policy parameters ϕ. The policy gradient theorem states that the gradient of the expected reward $J(\phi)$ is [58, Chapter 13.2]:

$$\nabla_\phi J(\phi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \nabla_\phi \log \pi \underbrace{\sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}}_{\Psi_t}\right], \quad (10)$$

where $\mathbb{E}_{\tau\sim\pi}[\square]$ denotes the expected value of $\square$ over a trajectory τ when following policy π and T is the time horizon of τ. Furthermore, $J(\phi)$ denotes the optimization objective of the policy, which is the expected reward in (10) and can be chosen differently depending on the used algorithm to tradeoff between bias and variance, resulting in a different form of Ψ_t, as shown in [59, Eq. 1].

In this work, we use the popular policy gradient algorithm Proximal Policy Optimization (PPO) [50] since it has demonstrated superior performance in continuous control problems compared to other state-of-the-art approaches [50]. Note that our safety layer also works with other RL methods for continuous action spaces. To improve training stability, one has to ensure that the policy is updated in a manner that it does not deviate too far from the previous policy π'. This is achieved in [60] by using a surrogate objective function that incorporates a trust-region constraint:

$$J(\phi) = \mathbb{E}_{\tau\sim\pi}\left[\eta(\phi)\hat{A}^{\pi'}(s_{\mathrm{RL}}, a_{\mathrm{RL}})\right] \quad (11)$$
$$\text{with } \eta(\phi) = \frac{\pi(a_{\mathrm{RL}}|s_{\mathrm{RL}})}{\pi'(a_{\mathrm{RL}}|s_{\mathrm{RL}})},$$

where $\hat{A}(\cdot)$ is the estimation of the true advantage function $A(\cdot)$, defined in [59, Eq. 3]. Although the update of policies is constrained, the training could still become unstable when the distance between the updated parameters ϕ and the previous parameters ϕ' is large. PPO solves this by forcing $\eta(\phi)$ to remain within a narrow range:

$$J(\phi) = \mathbb{E}_{\tau\sim\pi}\Big[\min(\eta(\phi)\hat{A}^{\pi'}(s_{\mathrm{RL}}, a_{\mathrm{RL}}), \mathrm{clip}(\eta(\phi), \epsilon)\hat{A}^{\pi'}(s_{\mathrm{RL}}, a_{\mathrm{RL}}))\Big], \quad (12)$$

where $\mathrm{clip}(\square, \epsilon)$ is an operator for constraining the operand $\square$ in the range $[1-\epsilon, 1+\epsilon]$.

We obtain $\hat{A}(\cdot)$ using the generalized advantage estimation proposed by Schulman et al. in [59], which combines multiple advantage estimators with different levels of bias and variance, resulting in a more robust estimator, thus providing more accurate and stable policy updates. The generalized advantage estimator is defined as:

$$\hat{A}_t = \sum_{i=0}^{T-t-1} (\gamma\lambda)^i \delta_{t+i}, \quad (13)$$
$$\text{with} \quad \delta_t = r_t + \gamma\hat{V}(s_{\mathrm{RL},t+1}) - \hat{V}(s_{\mathrm{RL},t}),$$

where λ is a hyperparameter that controls the tradeoff between bias and variance, and $\hat{V}(\cdot)$ is the estimation of the true value function $V(\cdot)$. In this work, we use an actor-critic structure [61] to approximate both the policy and value function with a shared neural network, which reduces variance of the estimated gradient of the expected reward and improves learning efficiency.

D. Problem Formulation

We consider the problem of computing a safe input trajectory for consecutive planning cycles. Without loss of generality, we set the initial time step of each planning cycle as $k = 0$. For each planning cycle, the problem is solved by projecting the actions proposed by the RL agent $u_{\mathrm{RL},[0,f]}$ to the closest safe actions $u_{[0,f]}$, such that the resulting trajectories are collision-free during the planning horizon $[t_0, t_f]$ and end within an invariably safe set at t_f. Note that we focus on ensuring legal safety in this paper, i.e., safety under the assumption that other traffic participants adhere to traffic laws. Given a collision-free initial state x_0 of the ego vehicle, we formulate our problem in a quadratic programming form [62, Ch. 4.4] for the longitudinal and lateral directions:

$$u_{[0,f-1]} = \arg\min \sum_{k\in[0,f-1]} \left\|u_k - u_{\mathrm{RL},k}\right\|^2, \quad (14)$$

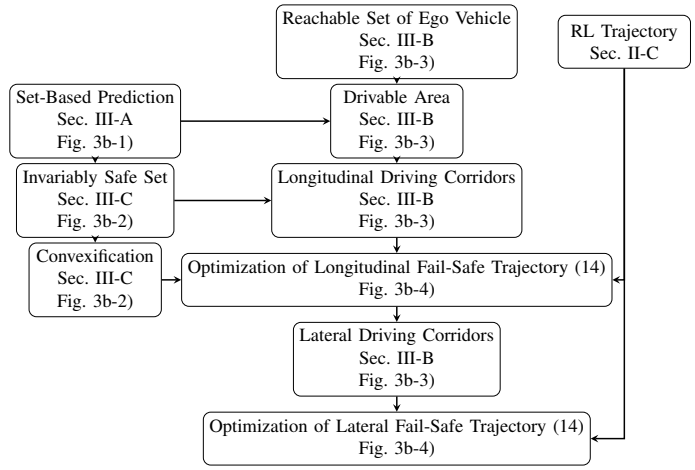
$$
\begin{aligned}
&\text{subject to} \\
&\forall k \in [0, f-1]: \quad x_{k+1} = A_k x_k + b_k u_k, \\
&\qquad\qquad\qquad\quad x_{k+1} \in \mathcal{X}_{\mathrm{ti},k+1}, \\
&\qquad\qquad\qquad\quad u_k \in \mathcal{U}_{\mathrm{ti}}, \qquad\qquad \text{(see II-B)} \\
&\qquad\qquad\qquad\quad x_{k+1} \in \mathcal{X}_{\mathrm{tv},k+1}, \qquad\qquad \text{(see III-B)} \\
&\text{for } k = f: \qquad x_k \in \mathcal{X}_{\mathrm{iss},k}, \qquad\qquad \text{(see III-C)}
\end{aligned}
$$

where $\mathcal{X}_{\mathrm{ti}}$ and $\mathcal{U}_{\mathrm{ti}}$ are the intervals originating from the linear constraints of the states and control inputs resulting from the system dynamics (2)-(8), $\mathcal{X}_{\mathrm{tv}}$ is the interval originating from the linear time-variant constraints obtained using reachability analysis, and $\mathcal{X}_{\mathrm{iss}}$ is the convexified under-approximation of the invariably safe set.

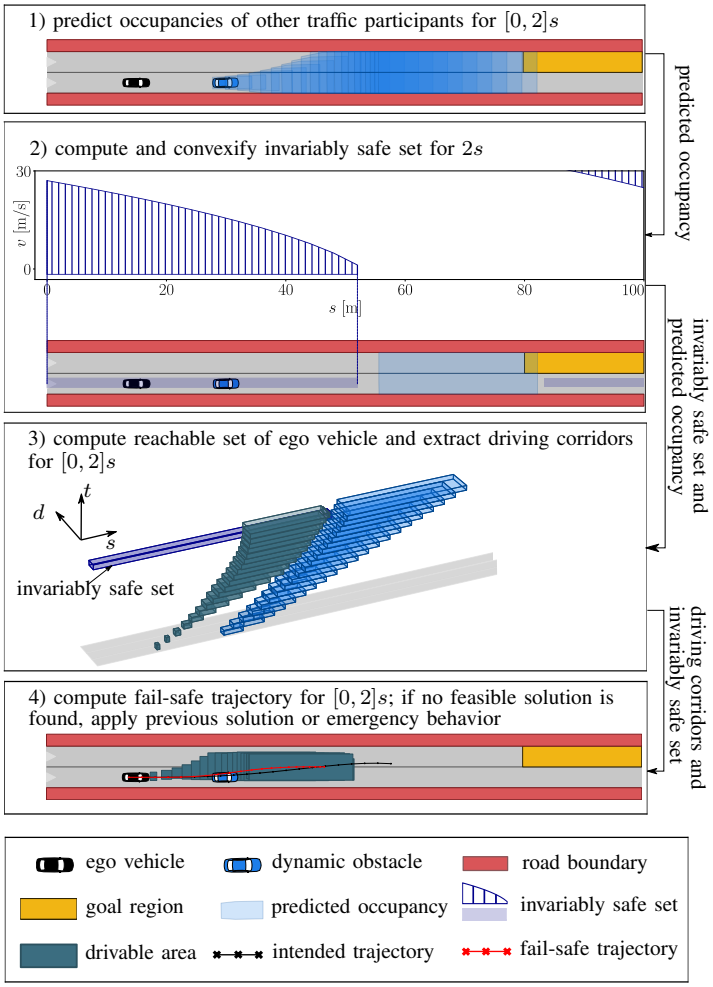
III. Solution Concept

For each planning cycle, we derive the safety constraints for (14) using the online verification framework illustrated in Fig. 3. Initially, we utilize reachability analysis based on physical constraints and traffic rules (implemented in [63]) to compute forbidden sets over time, i.e., the sets of all future motions of other traffic participants, see Fig. 3b-1). To ensure safety beyond the planning horizon, we compute an invariably safe set considering the forbidden sets and convexify the obtained set using polytopes, see Fig. 3b-2). Simultaneously, we compute the reachable sets of the ego vehicle by propagating its dynamics from a given initial set, remove the forbidden sets from the propagated reachable sets, and project them onto the position domain. This step allows us to obtain the collision-free drivable area for the ego vehicle, see Fig. 3b-3), which serves as the basis for extracting driving corridors (implemented in [64]). The longitudinal trajectory is then obtained by solving (14), utilizing the vehicle dynamics, the collision-free longitudinal driving corridors, and the convexified invariably safe sets as constraints. Subsequently, we extract the lateral driving corridors based on the optimized longitudinal trajectory. The difference between the longitudinal and lateral driving corridors is explained in Sec. III-B2. These lateral driving corridors are subsequently employed as time-variant constraints to solve equation (14) for determining the lateral trajectory, see Fig. 3b-4).

Because all our solutions are verified as legally safe, the automated vehicle never causes an accident. However, when the optimization problem (14) becomes infeasible, e.g., due to an illegal behavior of another vehicle, one could execute an emergency planner, which realizes collision mitigation.



(a) Flowchart of our online verification framework for one planning cycle.



(b) Exemplary computation steps for a simple scenario with a dynamic obstacle.

Fig. 3: Illustration of our online verification framework.

A. Set-Based Prediction for Other Traffic Participants

To predict the future motion of other traffic participants, we use the set-based prediction methods introduced in [54], [65], which account for uncertainties and offer tight over-approximations by considering traffic rules. Notably, compared to the worst-case predictions used in [11], [52], this approach uses consecutive time intervals instead of points in time, ensuring that no collisions occur between points in time.

To reduce the computational complexity involved in computing the reachable set of an exact model, abstractions are used. An abstraction is a simplified model, whose reachable set encloses the reachable set of the exact model. We list three abstractions used in this work in Table III, including the constraints they account for and reference proofs showing that their reachable sets are over-approximative. The final predicted

TABLE III: Abstractions used in set-based predictions.

Abstraction	Considered Constraints	Proof
acceleration-bounded abstraction	limits the absolute acceleration of a vehicle using a point-mass model	[65, Theorem 1]
lane-following abstraction	constrains the longitudinal motion: speed limit, forbidden reverse driving, limited engine power, allowed lanes	[65, Theorem 2]
safe-distance abstraction	certain vehicles have to maintain a safe distance to the ego vehicle	[54, Prop. 4]

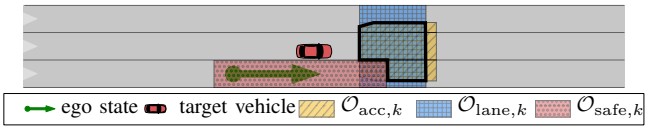


Fig. 4: An exemplary scenario showing the occupancies of three used abstractions, predicted for time 0.4s. The final predicted occupancy set $\mathcal{O}_k$ is marked with thick solid lines.

occupancy set of a dynamic obstacle is:

$$\mathcal{O}_k = \mathcal{O}_{\mathrm{acc},k} \cap \mathcal{O}_{\mathrm{lane},k} \cap \mathcal{O}_{\mathrm{safe},k}^{\complement}, \tag{15}$$

where $\mathcal{O}_{\mathrm{acc},k}$ denotes the occupancy of the acceleration and velocity-bounded abstraction, $\mathcal{O}_{\mathrm{lane},k}$ denotes the lane-following occupancy, and $\mathcal{O}_{\mathrm{safe},k}^{\complement}$ denotes the complement of the safe distance occupancy $\mathcal{O}_{\mathrm{safe},k}$, as depicted in Fig. 4. The intersected occupancies of all abstractions remain an over-approximation of the occupancy of the real model [65, Proposition 1]. We list all used parameters in Table VI. Note that the assumptions used in the abstractions are updated by updating the parameters (for $\mathcal{O}_{\mathrm{acc}}$ and $\mathcal{O}_{\mathrm{lane}}$) or deactivating the computation (for $\mathcal{O}_{\mathrm{safe}}$) as long as a violation is detected. The computational time of the set-based prediction method is linear in the number of traffic participants and prediction intervals [54, Sec. III].

B. Computation of Reachable Sets for the Ego Vehicle

After predicting traffic-rule-compliant future occupancies of other traffic participants, we need to determine the collision-free solution space to project the actions of the RL agent onto a safe area (14). It has been shown that using the reachable set of the ego vehicle to obtain constraints for an optimization-based motion planner [66] as well as to adjust the sampling intervals for a sampling-based motion planner [67] can effectively reduce computation time for complex traffic scenarios.

1) Procedure: To compute the reachable set of the ego vehicle, we adopt the polytopic set propagation method proposed in [66], [68] and implemented in the open-source toolbox CommonRoad-Reach [64]. As a set representation, this method uses Cartesian products of two polytopes as base sets $\mathcal{R}_k^{(i)}$to represent the reachable positions and velocities, in contrast to another method using polynomial zonotopes [42]. The advantage is that polytopes are closed under intersections and linear maps [69, Table 1], whereas the computational complexity raised by high-dimensional systems is irrelevant for our two-dimensional system. The computation of the reachable set involves the following steps [66, Sec. IV-C, Fig. 4]:

(i) Forward Propagation: The base sets $\mathcal{R}_k^{(i)}$ at time step k are propagated forward in time for the longitudinal and lateral direction using a double integrator vehicle model [66, Eq. 6] considering all admissible inputs. Note that this model over-approximates the friction circle, thus it is an abstract model. The resulting sets are denoted by $\mathcal{R}_{k+1}^{\mathrm{prop}}$. The initial base sets are created by enclosing the set of initial states $\mathcal{X}_0$ considering measurement errors, listed in Table VI.

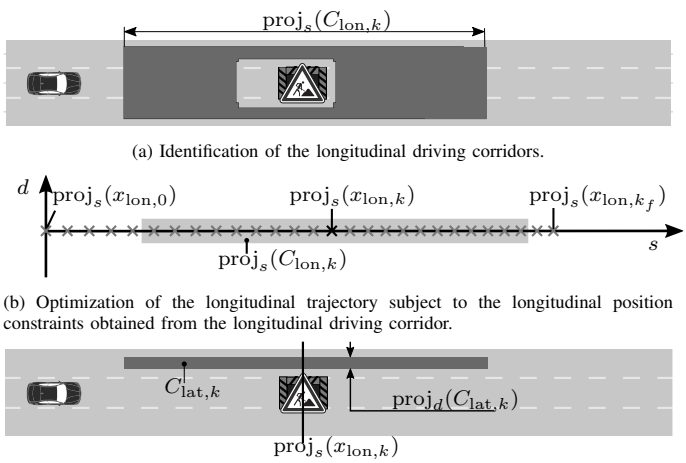
(ii) Collision Detection: To remove colliding states from $\mathcal{R}_{k+1}^{\mathrm{prop}}$, the set of forbidden occupancies $\mathcal{O}_{k+1}$ is first created using the union of the predicted occupancies obtained in Sec. III-A and road boundaries constructed using oriented rectangles [70, Sec. IV-B]. Next, we over-approximate $\mathcal{O}_{k+1}$ by the set of axis-aligned rectangles $\tilde{\mathcal{O}}_{k+1}$ to realize efficient set difference and union. The collision-free reachable set $\mathcal{R}_{k+1}$ is obtained by computing the difference between the union of the drivable area $\mathcal{D}_{k+1}^{\mathrm{prop}}$ and the union of $\tilde{\mathcal{O}}_{k+1}$, re-partitioning the resulting set into axis-aligned rectangles, and computing the reachable velocities for each collision-free drivable area. We over-approximate the shape of the ego vehicle using the three-circle approach proposed in [71] (see Fig. 2) and enlarge the drivable area accordingly when checking collisions as introduced in [66, Sec. IV-D, Appendix B].

(iii) Update of Reachability Graph: A directed graph $\mathcal{G}_{\mathcal{R}}$ is created for later extraction of driving corridors, where a node stores a base set $\mathcal{R}_k^{(i)}$ as well as its projection $\mathcal{D}_k^{(i)}$ and an edge $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)})$ stores the relation that $\mathcal{R}_{k+1}^{(j)}$ is reachable from $\mathcal{R}_k^{(i)}$ in one time step. The set of all edges are denoted by $\mathcal{E}$.

2) Driving Corridor and Constraint Extraction: After computing the drivable areas and the reachability graph, we need to extract the temporal sequence of the connected drivable areas, called driving corridors (denoted by C_{lon} and C_{lat}), from which we derive the spatiotemporal constraints for our optimization problem (14). We extract the longitudinal driving corridors that intersect with an invariably safe set $\mathcal{S}$ introduced in Sec. III-C to ensure safety beyond the planning horizon:

$$\begin{aligned} C_{\mathrm{lon}} := \big\{ &(\mathcal{D}_0^{(i_0)}, \mathcal{D}_1^{(i_1)}, ..., \mathcal{D}_f^{(i_f)}) \big| \\ &(\mathcal{R}_k^{(i_k)}, \mathcal{R}_{k+1}^{(i_{k+1})}) \in \mathcal{E}, \mathcal{D}_f^{(i_f)} \cap \mathcal{S}_f \neq \emptyset \big\}. \end{aligned} \tag{16}$$

Let $[d_{\min}, d_{\max}]$ denote the lateral bounds of the invariably safe set that the driving corridors intersect with and d_{des} denote the desired final lateral position resulting from the RL actions. We sort the extracted driving corridors by the distance between d_{des} and the closest lateral bound of the driving corridors, i.e., $\max(0, |d_{\mathrm{des}} - d_{\min}| - |d_{\mathrm{des}} - d_{\max}| - |d_{\max} - d_{\min}|)$. Note that we use zero when d_{des} is inside the driving corridor. Subsequently, we iterate over each longitudinal driving corridor, solve (14) for the longitudinal direction, and extract the lateral driving corridors according to the planned longitudinal positions, as visualized in Fig. 5c. The lateral driving corridors C_{lat} are obtained similarly as (16) except that sets $\mathcal{D}_k^{(i)}$ are



(a) Identification of the longitudinal driving corridors.

(b) Optimization of the longitudinal trajectory subject to the longitudinal position constraints obtained from the longitudinal driving corridor.

(c) Identification of the lateral driving corridors based on longitudinal planned positions.

Fig. 5: Illustration of extraction of driving corridors, adapted from [66, Fig. 3]

removed if they are not part of C_{lon} or do not enclose the planned longitudinal positions, see [66, Sec. V-C].

For each time step k, given the extracted longitudinal driving corridors, the longitudinal constraints are the interval between the lower and upper boundaries of the drivable areas:

$$\mathcal{X}_{\text{lon,tv},k} := \text{proj}_s(C_{\text{lon},k}) \times \mathbb{R}^3. \tag{17}$$

Let $d_{i,\min,k}, d_{i,\max,k}, i \in \{1,2,3\}$ denote the admissible lateral deviation of the centers of three circles that approximate the vehicle shape (see Fig. 2), where indices $\{1,2,3\}$ indicate the rear, center, and front circle, respectively; the lateral constraint for the rear circle is derived similarly using upper and lower boundaries of the drivable areas of the lateral driving corridors, i.e,:

$$[d_{1,\min,k}, d_{1,\max,k}] = \text{proj}_d(C_{\text{lat},k}). \tag{18}$$

For the derivation of the admissible lateral intervals of the center and front circles $d_{i,\min,k}, d_{i,\max,k}, i \in \{2,3\}$, see [66, Sec. VI-B2)]. Note that when the drivable areas are enlarged considering the vehicle shape during collision checks, the assumption of $e_\theta = 0$ is used, which is enforced by the way the admissible intervals are derived. Therefore, the enlargement of reachable sets considering the vehicle shape and extraction of lateral constraints have to be utilized together to ensure collision avoidance [66, Sec. VI-B2)]. Let l_{wb} denote the wheelbase of the vehicle. The lateral constraints are:

$$\mathcal{X}_{\text{lat,tv},k} := \Bigg\{ x_{\text{lat},k} \Bigg| \begin{bmatrix} d_{1,\min,k} \\ d_{2,\min,k} \\ d_{3,\min,k} \end{bmatrix} \leq \begin{bmatrix} 1 & 0 \\ 1 & \frac{1}{2} l_{\text{wb}} \\ 1 & l_{\text{wb}} \end{bmatrix} \begin{bmatrix} x_{\text{lat},k}^{(1)} \\ x_{\text{lat},k}^{(2)} \end{bmatrix} \leq \begin{bmatrix} d_{1,\max,k} \\ d_{2,\max,k} \\ d_{3,\max,k} \end{bmatrix} \Bigg\}. \tag{19}$$

C. Invariably Safe Sets

So far, the constraints in (17) and (19) ensure collision avoidance during the planning horizon. We compute invariably safe sets to ensure legal safety beyond the planning horizon, see Def. 5. To reduce the computational cost of obtaining the invariably safe set $\mathcal{S}_k$, Pek et al. [55] proposed an efficient way to compute a tight under-approximation of $\mathcal{S}_k$ using a safe distance set $\mathcal{S}_{\text{safe,k}}$[1]. The safe distance set considers a braking maneuver of the ego vehicle to avoid colliding with its preceding vehicle in case the preceding vehicle performs emergency braking.

1) Safe Distance Set: Before presenting our algorithm, we first introduce the used notation: The vector of parameters of the ego vehicle is denoted by $\rho_{\text{ego}} := [a_{\text{lat,max}}, a_{\text{lon,max}}, v_{\max}, w, l_{\text{front}}, l_{\text{rear}}, \delta_{\text{react}}]$, where $a_{\text{lat,max}}$ and $a_{\text{lon,max}}$ are the maximal accelerations in the lateral and longitudinal direction, respectively; $v_{\max}$ is the maximal velocity; w is the width; l_{front} and l_{rear} are the distances between the rear axle center and the front and rear edge of the ego vehicle, respectively; and δ_{react} is the reaction time of the ego vehicle. The set of all reachable lanes that lead to the goal region of the ego vehicle is represented by $\mathcal{L}$. We assume that the curvature $\kappa(s)$ as well as the lateral position of the left and right boundaries $d_{\text{lb,left}}(s), d_{\text{lb,right}}(s)$ for position s is accessible from $\mathcal{L}$. Let $\mathcal{C}_{i,j}$ denote a collision-free section (see Fig. 6) on a lane of a pair of vehicles (b_i, b_j), where i and j are indices of the succeeding and preceding vehicle, respectively; we compute the safe distance set for a collision free section $\mathcal{S}_{\text{safe},k}^{\mathcal{C}_{i,j}}$, as shown in Algorithm 1 (see Appendix B for the difference to [55, Alg. 1]).

Algorithm 1 safeDistanceSetForSection(), modified from [55]

Input: k, ρ_{ego}, $\mathcal{C}_{i,j}, \mathcal{O}_{i,k}, v_{i,k}, \mathcal{O}_{j,k}, v_{j,k}, \kappa, d_{\text{lb,left}}, d_{\text{lb,right}}$
Output: Safe distance set of a section $\mathcal{S}_{\text{safe},k}^{\mathcal{C}_{i,j}}$

 # Acceleration constraints [72, Eq. 2-4]
1: $r_{\min} = \min(1/|\kappa(\mathcal{C}_{i,j})|)$
2: $v_{\text{crit}} = \sqrt{r_{\min} a_{\text{lat,max,ego}}}$
3: $a_{\text{lon}}(v) = a_{\text{lon,max,ego}} \sqrt{1 - (v^2/v_{\text{crit}}^2)^2}$
 # Safe distance to preceding vehicle b_j [73, Eq. 17]
4: $\zeta(v, b_j) = v^2/2|a_{\text{lon}}(v)| - v_j^2/2|a_{\text{lon,max},j}| + \delta_{\text{react}} v$
5: $\Delta_{\text{safe}}^k(v, b_j) = \max\big(\zeta(v, b_j), 0\big)$
 # Safe distance to succeeding vehicle b_i [73, Eq. 17]
6: $\zeta(v, b_i) = v_i^2/2|a_{\text{lon,max},i}| - v^2/2|a_{\text{lon}}(v)| + \delta_{\text{react}} v_i$
7: $\Delta_{\text{safe}}^k(v, b_i) = \max\big(\zeta(v, b_i), 0\big)$
 # Lateral constraint:
8: $d_{\text{lim}}(\mathcal{C}_{i,j}) = r_{\min} - \sqrt{r_{\min}^2 - l_{\text{front}}^2} + 0.5w$, see Fig. 6
9: $d_{\min} = \max(d_{\text{lb,right}}(\mathcal{C}_{i,j})) + d_{\text{lim}}(\mathcal{C}_{i,j})$
10: $d_{\max} = \min(d_{\text{lb,left}}(\mathcal{C}_{i,j})) - d_{\text{lim}}(\mathcal{C}_{i,j})$
 # Safe distance set of a section $\mathcal{C}_{i,j}$:
11: $\mathcal{S}_{\text{safe},k}^{\mathcal{C}_{i,j}} \leftarrow \{(s, v, d)^T \in \mathcal{X} \,|\, d_{\min} \leq d \leq d_{\max}$
$\wedge v \leq \min\big(v_{\text{crit}}, v_{\max}\big) \wedge s \in \mathcal{C}_{i,j}$
$\wedge s \geq \max(\text{proj}_s(\mathcal{O}_{i,k})) + \Delta_{\text{safe}}^k(v, b_i)$
$\wedge s \leq \min(\text{proj}_s(\mathcal{O}_{j,k})) - \Delta_{\text{safe}}^k(v, b_j)\}$
12: **return** $\mathcal{S}_{\text{safe},k}^{\mathcal{C}_{i,j}}$

Given $\mathcal{L}$, ρ_{ego}, the set of predicted occupancies and velocity intervals of all safety-relevant traffic participants $\{(\mathcal{O}_{b,k}, v_{b,\min,k}, v_{b,\max,k}) | \, b \in \mathcal{B}\}$, and the longitudinal position of the ego vehicle at the current time step $s_{\text{ego},0}$, the safe distance set $\mathcal{S}_{\text{safe,k}}$ at time step k is computed as in Algorithm 2. In Line 1-16, we iterate over each lane and

[1]The work in [55] also considered a set of safe evasive maneuvers. To keep the presentation simple, we intentionally do not consider this set, as its effect was limited in the considered scenarios, see Appendix A.

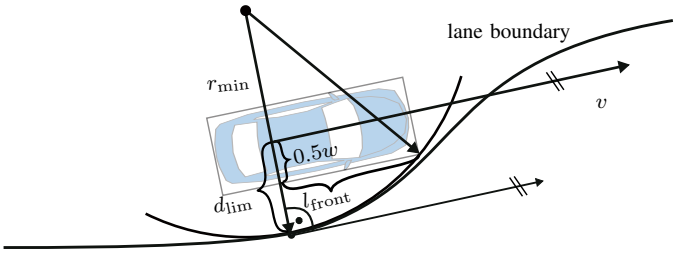


Fig. 6: Limit of lateral position of the reference point $d_{\lim}$ before the vehicle starts to intersect with another lane.

compute the safe distance set between each vehicle pair in this lane accordingly. Furthermore, when the occupancy of the ego vehicle intersects with two adjacent lanes, the ego vehicle needs to consider safe distances to vehicles in both lanes, thus in Line 17-24, we compute the set of (s, v) as the intersection between the safe distance set and the lateral constraints considering the limits when the ego vehicle starts to intersect with both lanes. We show an example of the computed safe distance sets of two lanes and the intersections of projected sets in the $s - v$ domain in Fig. 7.

Algorithm 2 safeDistanceSet()

Input: $k, \rho_{\text{ego}}, s_{\text{ego},0}, \{(\mathcal{O}_{b,k}, v_{b,\min,k}, v_{b,\max,k}) | \, b \in \mathcal{B}\}, \mathcal{L}$
Output: Safe distance set $\mathcal{S}_{\text{safe,k}}$

1: Initialize $\mathcal{S}_{\text{safe,k}} \leftarrow \emptyset$, $\mathcal{C} \leftarrow \emptyset$
 # Compute safe distance set for each lane
2: **for** $\texttt{lane} \in \mathcal{L}$ **do**
3: Initialize $\mathcal{S}^{\texttt{lane}}_{\text{safe},k} \leftarrow \emptyset$
4: $B^{\texttt{lane}} \leftarrow [\, b \in \mathcal{B} \,|\, \mathcal{O}_{b,k} \cap \texttt{lane} \neq \emptyset \,]$
5: $B^{\texttt{lane}} \leftarrow B^{\texttt{lane}}$ sorted by $\inf\{\text{proj}_s(\mathcal{O}_{b,k})\}$
6: **for** $i = 1, 2, ..., |B^{\texttt{lane}}| - 1$ **do**
7: $b_i \leftarrow B^{\texttt{lane}}[i]$, $b_j \leftarrow B^{\texttt{lane}}[i+1]$
8: $\mathcal{C}_{i,j} \leftarrow [\max(\text{proj}_s(\mathcal{O}_i)) + l_{\text{front}}, \min(\text{proj}_s(\mathcal{O}_j)) - l_{\text{rear}}], \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{i,j}$
9: **if** $s_{\text{ego},0} \in \mathcal{C}_{i,j}$ **then**
10: $b_i \leftarrow \texttt{null}, \mathcal{O}_{i,k} \leftarrow \emptyset, v_{i,\max,k} \leftarrow \texttt{null}$
11: **end if**
12: $\mathcal{S}^{\mathcal{C}_{i,j}}_{\text{safe},k} \leftarrow$ safeDistanceSetForSection($k, \rho_{\text{ego}}, \mathcal{C}_{i,j}, \mathcal{O}_{i,k}, v_{i,\max,k}, \mathcal{O}_{j,k}, v_{j,\min,k}, \kappa, d_{\text{lb,left}}, d_{\text{lb,right}}$)
13: $\mathcal{S}^{\texttt{lane}}_{\text{safe},k} \leftarrow \mathcal{S}^{\texttt{lane}}_{\text{safe},k} \cup \mathcal{S}^{\mathcal{C}_{i,j}}_{\text{safe},k}$
14: **end for**
15: $\mathcal{S}_{\text{safe,k}} \leftarrow \mathcal{S}_{\text{safe,k}} \cup \mathcal{S}^{\texttt{lane}}_{\text{safe},k}$
16: **end for**
 # Intersect safe distance sets of each pair of adjacent lanes
17: **for** $\mathcal{C}_{i,j} \in \mathcal{C}$ **do**
18: **for** $\mathcal{C}_{m,n} \in$ leftAdjacentSections($\mathcal{C}_{i,j}$) **do**
19: $d_{\min} \leftarrow \min(d_{\text{lb,left}}(\mathcal{C}_{i,j})) - d_{\lim}(\mathcal{C}_{i,j})$
20: $d_{\max} \leftarrow \max(d_{\text{lb,left}}(\mathcal{C}_{i,j})) + d_{\lim}(\mathcal{C}_{i,j})$
21: $\mathcal{S}^{\texttt{int}}_{\text{safe},k} \leftarrow \text{proj}_{s,v}(\mathcal{S}^{\mathcal{C}_{i,j}}_{\text{safe},k}) \cap \text{proj}_{s,v}(\mathcal{S}^{\mathcal{C}_{m,n}}_{\text{safe},k}) \times [d_{\min}, d_{\max}]$
22: $\mathcal{S}_{\text{safe,k}} \leftarrow \mathcal{S}_{\text{safe,k}} \cup \mathcal{S}^{\texttt{int}}_{\text{safe},k}$
23: **end for**
24: **end for**
25: **return** $\mathcal{S}_{\text{safe,k}}$

2) Convexification of Safe Distance Set: As depicted in Fig. 7, the obtained $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$ might be non-convex when a succeeding vehicle is present. In order to utilize convex optimization to solve (14), it is necessary to convexify $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$. Since $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$ is computed using the safe distances (see Line 4 and Line 6 of Alg. 1), only the curved part of the lower bound could cause non-convexity of $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$. Therefore, we develop a simple method to determine a convex under-approximation of $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$. As shown in Fig. 7, we find the curved part of the lower bound which causes the non-convexity, obtain a straight line (see dashed line in Fig. 7) passing through the first point (s_0, v_0) and last point (s_n, v_n) of the curved part, and shift the straight line until it does not intersect with the curved part. A detailed implementation of this method is given in Appendix C.

An example of the convexified sets is illustrated in Fig. 7. Note that our method only works for convexifing the safe distance set. If the evasive set is used as a constraint, one could deploy a more general method such as iterative regional inflation by semidefinite programming (iris) [74], which iteratively inflates regions within a non-convex area until a sufficiently large polytope is obtained. We compare the conservativeness of iris and our method for the safe distance set empirically in Table IV and conclude that our method is less conservative and 85 times faster.

We use the halfspace representation $\langle A_{\text{iss}}, b_{\text{iss}} \rangle_{\tilde{\mathcal{P}}}$ for the approximated polytope $\tilde{\mathcal{P}}$, then the longitudinal constraint derived from the safe distance set for time step k is

$$\mathcal{X}_{\text{lon,iss,k}} := \langle A_{\text{iss}}, b_{\text{iss}} \rangle_{\tilde{\mathcal{P}}} \times \mathbb{R}^2 \tag{20}$$

and the lateral constraint from the lateral limits $d_{\min}$ and $d_{\max}$ of $\mathcal{S}_{\text{safe,k}}$ is:

$$\mathcal{X}_{\text{lat,iss,k}} := [d_{\min}, d_{\max}] \times [0, 0, 0]^T. \tag{21}$$

Note that constraints (20) and (21) are used for the last time step of the planning horizon, as shown in (14).

IV. Numerical Experiments

A. Setting

1) General Setting: We train and evaluate all agents in the open-source environment CommonRoad-RL [75]. All training uses the RL algorithm PPO [50] detailed in Sec. II-C. We adopt the implementation of PPO from Stable Baselines[2] and list the hyperparameters of the policy and value networks in Table VI. For all experiments, we split the available traffic scenarios into 70% training scenarios and 30% test scenarios: for each group of experiments, we show the learning curves for the training scenarios and evaluate the trained agents using the test scenarios. We first compare different vehicle models without the safety layer in Sec. IV-B. Next, we evaluate the effectiveness of the proposed safety layer in Sec. IV-D for randomly sampled actions and for training an RL agent. All experiments are carried out on a server with an AMD EPYC 7742 2.2 GHz processor.

[2] github.com/hill-a/stable-baselines

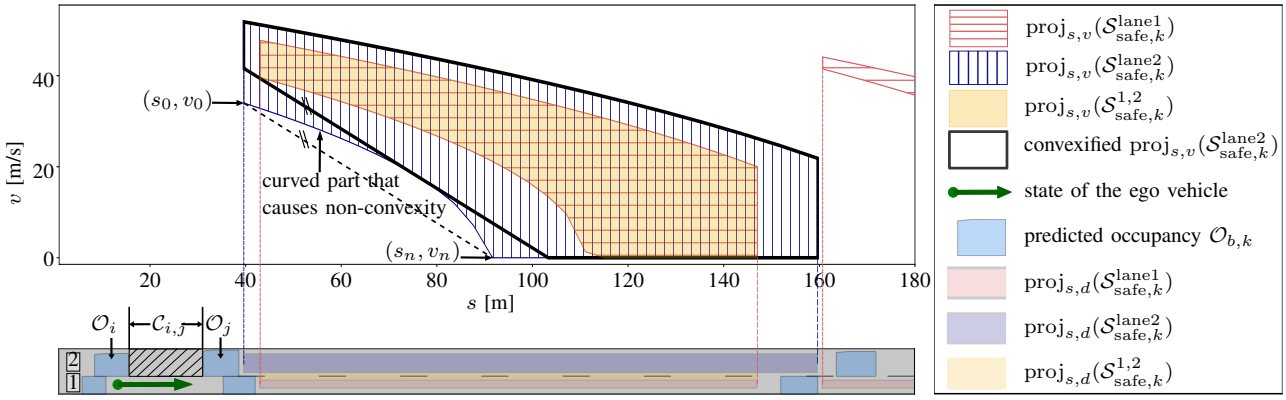


Fig. 7: An example of safe distance sets of two lanes and their intersections projected in $s-v$ domain. Note that we only predict occupancies of the dynamic obstacles in their current lanes for illustration purposes.

2) Dataset: We evaluate the proposed approach using the highD dataset [76], which consists of real-world highway drone data capturing naturalistic vehicle trajectories. The dataset was collected at six locations on German highways featuring two-lane and three-lane roads. Spanning over 16.5 hours and covering a distance of more than 45,000 kilometers, the highD dataset provides vehicle trajectories with a time resolution of $\Delta t = 0.04\,\text{s}$. To facilitate the evaluation process, we transform the dataset into 2816 individual CommonRoad scenarios [77] using an open-source converter[3], each lasting 40 seconds. We randomly select a vehicle for each scenario and formulate a planning problem based on its initial and final states. Subsequently, we remove the selected vehicle from the scenario to simulate its absence during planning and analysis.

3) Observations: We use the observations defined in [75, Tab. II], including the states of the ego vehicle, topological information of the road network, information of the goal region, and states of the surrounding traffic participants using the lane-based surrounding detection. We attach the used observations in Appendix D for reachability.

4) Actions: RL often works as a step-by-step decision maker, i.e., the policy network receives the observations from the environment every time step and only outputs the control input of the system for one time step. However, motion planners of automated vehicles need to plan a trajectory for a time horizon to account for processing and computation times. Furthermore, to use our online verification framework as the safety layer, we need to generate an intended trajectory of actions. Therefore, we let the policy network output the actions at time step k for the control inputs for time horizon $[k, k+f]$:

$$a_{\text{RL},k} := \begin{bmatrix} u_{\text{lon},[k,k+f]} \\ u_{\text{lat},[k,k+f]} \end{bmatrix}. \tag{22}$$

5) Reward Functions: For the reward function, we chose the sparse reward and dense reward proposed by [75, Sec. III-D]. Before defining the rewards, we first introduce the following binary variables:

- $\mathbf{1}_{\text{reach\_goal}} = 1$ 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.

The sparse reward is defined as:

$$r_{\text{sparse}} = r_{\text{reach\_goal}} + r_{\text{collision}} + r_{\text{off\_road}} + r_{\text{time\_out}}, \tag{23}$$

where $r_{\square} = c_{\square}\mathbf{1}_{\square}$ and $c_{\square}$ denotes coefficients for each partial reward. In addition, the dense reward is defined as:

$$r_{\text{dense}} = r_{\text{sparse}} + r_{\text{closer}} + r_{\text{safe\_dist}} + r_{\text{road\_center}}, \tag{24}$$

where

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

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 step k, respectively. Furthermore, d_{lead} and d_{safe} denote the current distance and safe distance [78] 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.

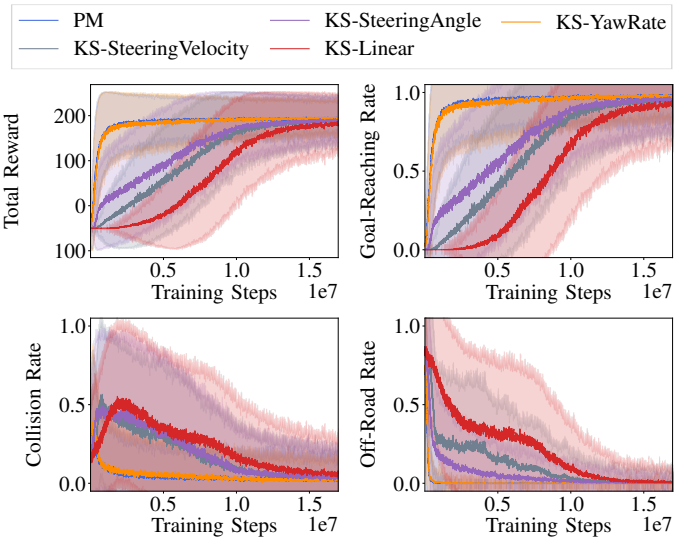
We compared both rewards for all our numerical experiments, and they did not show a significant difference regarding the convergence. Therefore, we only show the results of the agents using the sparse reward subsequently.

B. Evaluation of Unsafe Agents

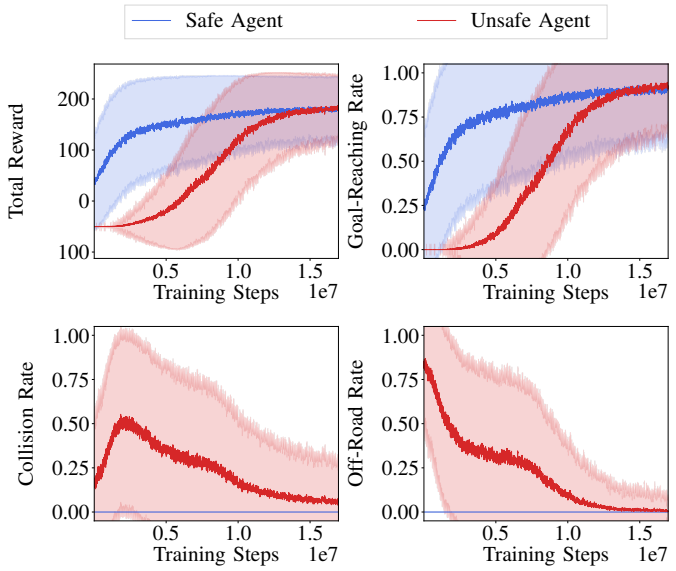
We compare different vehicle models available in [75] when using input trajectories as actions. In addition, the states of each vehicle model are added in the observation space. We introduce the following abbreviations for the vehicle models:

- **PM**: denotes the point-mass model, which ignores the non-holonomic behavior of the vehicle [77, Sec. III-A];
- **KS-SteeringAngle**: denotes the kinematic single-track model, which uses the acceleration and normalized steering angle as inputs [79, Sec. VII];
- **KS-SteeringVelocity**: denotes the kinematic single-track model, which uses the acceleration and steering velocity as inputs [77, Sec. III-B];

[3]commonroad.in.tum.de



(a) Learning curves of different vehicle models returning a trajectory of actions. Note that the result of PM overlaps with the curves of KS-YawRate.

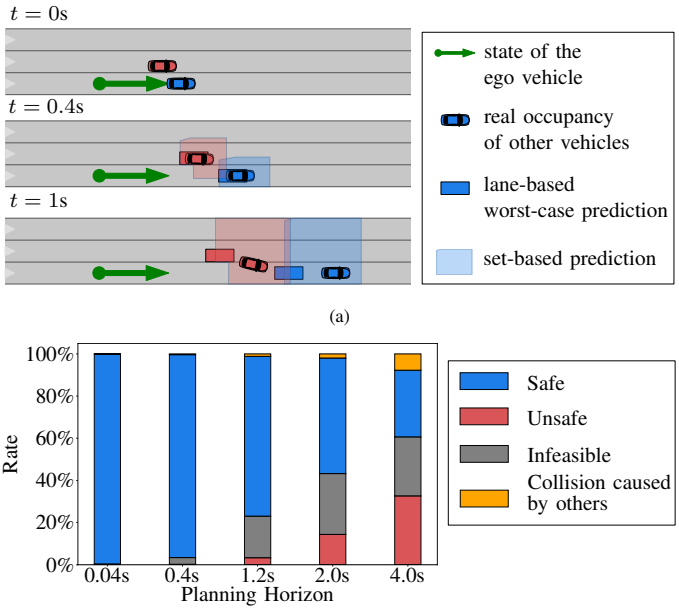


(b) Learning curves of the unsafe agent and safe agent.

Fig. 8: Learning curves for highD training scenarios.

- **KS-YawRate**: denotes the kinematic single-track model which uses the acceleration and yaw rate as inputs [11, Sec. II-A];
- **KS-Linear**: the linearized kinematic single-track model used in this paper, as introduced in Sec. II-B. Note that this model has the highest order among all used models, thus is the closest to a high-fidelity model and has been applied in real test drives in [56, Sec. VI].

We show the learning curves of different vehicle models for the training scenarios in Fig. 8a, from which we can conclude that KS-Linear needs the most training steps to converge due to its more complicated dynamics, which requires more interaction with the environment to implicitly learn the dynamics. During the first one-third of the training, KS-Linear almost always crashed or drove offroad due to random exploration, thus constantly receiving negative rewards. This motivates the use of our safety layer since it prevents the agent from colliding with other obstacles or driving offroad.



(a)

(b)

Fig. 9: a) An exemplary scenario comparing the set-based predictions using reachability analysis [54] and the lane-based worst-case assumption in [11] that all preceding vehicles brake with maximal deceleration. b) Evaluation of the control barrier function method [11] for multiple planning horizons.

C. Comparison with Related Work

1) Comparison of Set-Based Prediction and Worst-Case Assumption: We compare our set-based prediction and the lane-based worst-case assumption used in [11] that the preceding vehicles of the ego vehicle brake with the maximal deceleration at different times for an exemplary highway scenario in Fig. 9a. Both predictions are safe regarding the ego vehicle when the prediction horizon is short (0.4s). However, for a longer prediction horizon (1s), the lane-based worst-case assumption does not account for the lane change of the red vehicle, while the set-based prediction contains its actual occupancy, which maintains a safe distance to the ego vehicle, and thus still complies with traffic rules.

2) Evaluation of the Control Barrier Function Method: As the accuracy of the prediction used in [11] decreases with increasing prediction horizons, safety could also be compromised when the planning horizon increases. To validate this conjecture, we evaluate the control barrier function method used in [11] using randomly sampled actions for multiple planning horizons. Note that in the optimization problem [11, Eq. 11], the constraints of u depend on the state of the system x, thus only when solving for one time step, [11, Eq. 11] is a convex problem by assuming x to be constant within one time step. To increase the planning horizon while keeping the optimization convex, we solve [11, Eq. 11] in a model predictive fashion, i.e., by estimating x using the dynamics of the system. We summarize reasons that a scenario is terminated in Fig. 9b, where *safe* means if the agent reaches the goal or times out, *unsafe* means when the agent causes a collision with other obstacles or goes offroad, *infeasible* represents the cases that the relaxed optimization problem is still infeasible. As shown in Fig. 9b, the unsafe rate increases with increasing planning horizon. In addition, we evaluate our

TABLE IV: Evaluation of convexified safe distance set using highD scenarios.

Method	δ_{iss} mean $\pm$ std..	Avg. Comput. Time
iris [74]	0.907 ± 0.059	$185.59\,\text{ms}$
ours	0.916 ± 0.069	$2.17\,\text{ms}$

TABLE V: Evaluation of safety layer using randomly sampled actions.

Termination Reason	Rate
infeasible initial state - longitudinal	16.48%
infeasible initial state - lateral	1.13%
collision caused by other vehicles	11.51%
goal area reached	23.79%
time out	47.09%
Module	**Avg. Comput. Time [ms]**
safety layer - total	473.25
set-based prediction	184.77
invariably safe set	42.53
longitudinal driving corridor	107.37
lateral driving corridor	4.10
longitudinal optimization	43.95
lateral optimization	67.67

method using randomly sampled actions for multiple planning horizons as well, where the unsafe rate is always zero.

D. Evaluation of the Safety Layer

In our evaluation, when the optimization problem (14) becomes infeasible, e.g., due to an illegal behavior of another vehicle, we continue following the previous solution. When we reach the end of the previous solution and still cannot find a feasible trajectory, we switch to an emergency planner. Note that due to convexification of the invariably safe set, a safe emergency behavior could still exist even when (14) is infeasible. To ease the reproduction of our method, we use a simple method as the emergency planner. Since the invariably safe set is determined by considering the safe distance to both the preceding and succeeding vehicles, the appropriate safe action should be either full braking to avoid collision with the preceding vehicle or full acceleration to avoid collision with the succeeding vehicle. Therefore, when the safe distance to the ego vehicle is violated, full braking will be performed; otherwise, full acceleration will be performed.

1) Evaluation of Convexified Safe Distance Set: We evaluate the convexified set $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$ using our method introduced in Sec. III-C2 and iris [74] for the highD dataset. As an evaluation metric for the conservativeness, we utilize the ratio of the area of the approximated polytope $\tilde{\mathcal{P}}$ and original polygon $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$:

$$\delta_{\text{iss}} = \frac{\text{area of } \tilde{\mathcal{P}}}{\text{area of } \text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})}. \tag{25}$$

In addition, we show the average computation time for both methods in Table IV, from which we can conclude that our method is less conservative and 85 times faster.

2) Random Actions: To ensure that all scenarios used during training and evaluation are feasible, we first evaluate all scenarios using the safety layer by simulating the ego vehicle using randomly sampled actions. We summarize the termination reasons of all scenarios and the average computation time of each safety module in Table V. Since our software is modularized for quick prototyping, the runtime of the implementation could be further improved by eliminating the Python interfaces and using only the core C++ code for all computations. We remove the scenarios for which the initial states are infeasible. Note that since the lateral optimization problem depends on the solution of the longitudinal optimization problem, an infeasible lateral initial state could still become feasible if a different longitudinal trajectory is used. Thus only the scenarios with initial states that are infeasible in the longitudinal direction are removed, resulting in 2352 scenarios.

3) Safe RL Agents: We train an agent with the safety layer using the initially feasible scenarios and compare the learning curve with the unsafe agent in Fig. 8b. During training, the safe agent has never caused a collision with other obstacles or gone offroad. In addition, with the help of the safety layer, the safe agent converged much faster than the unsafe agent, demonstrating the effectiveness of our method.

V. Conclusions

This article presents an approach to provide safety guarantees for model-free RL motion planners in real-time for infinite planning horizons. Our proposed integration of online reachability analysis with model-free RL can handle arbitrary traffic situations and outperforms another safety method based on control barrier functions for longer planning horizons. Numerical experiments using real-world traffic data demonstrate the effectiveness of our framework in complex and unpredictable traffic situations without compromising the performance. Our work thus sets a foundation for future explorations and improvements in the area of safe reinforcement learning.

Appendix

A. Evaluation of Safe Distance Set and Evasive Distance Set

An evasive distance set $\mathcal{S}_{\text{eva,k}}$ considers the evasive maneuver that the ego vehicle swerves to an adjacent lane to avoid a collision [55]. In this work, we only consider the safe distance set $\mathcal{S}_{\text{safe,k}}$ and omit the computation of $\mathcal{S}_{\text{eva,k}}$ for the following reasons:

- Computing $\mathcal{S}_{\text{eva,k}}$ requires us to additionally predict the occupancy set of the obstacles for a time horizon $[k, k + t_{\text{eva}}]$, where t_{eva} denotes the time steps required for the evasive maneuver. Because the ego vehicle has to maintain a safe distance from the obstacles on the target lane after the evasive maneuver, see $\mathcal{S}_1^{t+r}$ in [55, Alg.1 Line 11]. This adds computational burden of $\mathcal{O}(|\mathcal{B}|t_{\text{eva}})$;
- As our evaluation in Table VII shows, computing $\mathcal{S}_{\text{eva,k}}$ does not provide a sufficiently larger solution space compared to $\mathcal{S}_{\text{safe,k}}$;
- The safe distance set $\mathcal{S}_{\text{safe,k}}$ is still an under-approximation of $\mathcal{S}_k$, i.e., $\mathcal{S}_{\text{safe,k}} \subset \mathcal{S}_k$. This is because $\mathcal{S}_{\text{safe,k}} \subseteq \mathcal{S}_{\text{safe,k}} \cup \mathcal{S}_{\text{eva,k}}$ and $\mathcal{S}_{\text{safe,k}} \cup \mathcal{S}_{\text{eva,k}} \subset \mathcal{S}_k$ [55, Proposition 1].

Since computation of the evasive distance set adds computational costs, we aim to evaluate the necessity of computing the evasive distance set by comparing the resulting size of the safe distance set and the evasive distance set. We compute the areas of $\text{proj}_{s,v}(\mathcal{S}_{\text{safe,k}})$ and $\text{proj}_{s,v}(\mathcal{S}_{\text{eva,k}})$ for all highD Scenarios

TABLE VI: Parameters used in the numerical experiments.

Notation	Meaning	Value
Planning Parameters		
Δt	time step size	$0.04\,\mathrm{s}$
t_{plan}	replanning frequency	$0.4\,\mathrm{s}$
t_f	planning horizon	$2\,\mathrm{s}$
Vehicle Parameters		
$a_{\max}$	maximal absolute acceleration	$11.5\,\mathrm{m/s^2}$
$a_{\mathrm{lon,max}}$	maximal longitudinal acceleration	$11.5\,\mathrm{m/s^2}$
$a_{\mathrm{lat,max}}$	maximal lateral acceleration	$11.5\,\mathrm{m/s^2}$
$a_{\mathrm{comfort}}^{\mathrm{ego}}$	acceleration of ego vehicle when computing safe distance in SPOT	$4\,\mathrm{m/s^2}$
$v_{\min}$	minimal longitudinal velocity	$0\,\mathrm{m/s}$
$v_{\max}$	maximal absolute velocity	$65\,\mathrm{m/s}$
v_S	switching velocity	$5\,\mathrm{m/s}$
$j_{\min}, j_{\max}$	minimal/maximal jerk	$\pm 10\,\mathrm{m/s^3}$
$\ddot{a}_{\min}, \ddot{a}_{\max}$	minimal/maximal jounce	$\pm 10^3\,\mathrm{m/s^4}$
$e_{\theta,\min}, e_{\theta,\max}$	minimal/maximal relative orientation	± 0.1408
$\kappa_{\min}, \kappa_{\max}$	minimal/maximal curvature	$\pm 0.2\,/\mathrm{m}$
$\dot{\kappa}_{\min}, \dot{\kappa}_{\max}$	minimal/maximal derivative of curvature	$\pm 0.2\,/\mathrm{m\,s}$
$\ddot{\kappa}_{\min}, \ddot{\kappa}_{\max}$	minimal/maximal second derivative of curvature	$\pm 0.4\,/\mathrm{m\,s^2}$
f_{speed}	speeding factor	1.2
δ_{react}	reaction delay of ego vehicle when computing safe distance	$0.3\,\mathrm{s}$
w	width of ego vehicle	$1.61\,\mathrm{m}$
l_{wb}	wheelbase of ego vehicle	$2.579\,\mathrm{m}$
l_{front}	distance between rear axle center and front edge of ego vehicle	$3.543\,\mathrm{m}$
l_{rear}	distance between rear axle center and rear edge of ego vehicle	$0.965\,\mathrm{m}$
Reachable Set Parameters		
Δs_0	uncertainty initial longitudinal position	$0.1\,\mathrm{m}$
Δd_0	uncertainty initial lateral position	$0.1\,\mathrm{m}$
$\Delta v_{\mathrm{lon},0}$	uncertainty initial longitudinal velocity	$0.1\,\mathrm{m/s}$
$\Delta v_{\mathrm{lat},0}$	uncertainty initial lateral velocity	$0.1\,\mathrm{m/s}$
Hyperparameters and Neural Network Structure of PPO		
λ	tradeoff factor between bias and variance in (13)	0.95
γ	discount factor	0.99
ϵ	clip range in (12)	0.2
-	number of epochs	10
-	minibatch size	32
-	learning rate	0.0005
-	actor-critic architecture	shared networks
-	policy/value network type	MLP
-	network structure	[64, 64]
-	activation function	tanh

TABLE VII: Comparison of safe distance set and evasive distance set.

Description	Area $\mathrm{proj}_{s,v}(\mathcal{S}_{\mathrm{safe,k}})$	Area $\mathrm{proj}_{s,v}(\mathcal{S}_{\mathrm{eva,k}})$
mean	58870.00	51064.26
std.	6734.57	6333.84

by simulating the ego vehicle using randomly sampled actions and list the results in Table VII. Since the safe distance set is larger than the safe evasive set on average, we can conclude that computing $\mathrm{proj}_{s,v}(\mathcal{S}_{\mathrm{eva,k}})$ does not provide a much larger solution space. Thus we omit its computation during training.

B. Modification of Safe Distance Sets for Sections

Algorithm 1 adds the following modifications to [55, Alg. 1]:

- For each $\mathcal{C}_{i,j}$, [55, Algorithm 1] only computes the safe distance to the preceding vehicle b_j, whereas we also take into account the succeeding vehicle in Line 5-7, except for the section where the ego vehicle is currently located in.
- We compute $\mathcal{S}_{\mathrm{safe,k}}$ for the rear axle center of the vehicle, instead of the geometric center of the vehicle used in [55], because we use the rear axle center of the vehicle as the reference point. Therefore, we enlarge $\mathrm{proj}_s(\mathcal{O}_j)$ by l_{front} and $\mathrm{proj}_s(\mathcal{O}_i)$ by l_{rear}, see Line 8.
- We add the lateral constraints considering the lateral dimension of the ego vehicle in Line 7-10. Note that when computing the safe distance [73, Eq. 17], it is assumed that the ego vehicle can brake with the maximal available acceleration along a reference path. In other words, the safe distance set is only valid if the ego vehicle aligns with the reference path. Therefore, we enforce this assumption by using $e_{\theta,f} = \kappa_f = \dot{\kappa}_f = 0$, see (21). Consequently, the limit of d before the ego vehicle starts to intersect with another lane can be computed as Line 8, as shown in Fig. 6. Note that the minimum turning radius of passenger cars is $7.3\,\mathrm{m}$[4],thus we assume $r_{\min} > l_{\mathrm{front}}$ to ensure validity of $\sqrt{r_{\min}^2 - l_{\mathrm{front}}^2}$ in Line 8.

C. Algorithm of Convexification of Safe Distance Set

Algorithm 3 convexSafeDistanceSet()

Input: $\mathrm{proj}_{s,v}(\mathcal{S}_{\mathrm{safe,k}})$
Output: $\langle A_{\mathrm{iss}}, b_{\mathrm{iss}} \rangle_{\tilde{\mathcal{P}}}$

 # Find upper bound and the curved part of lower bound
1: $[v_{\min}, v_{\max}] := \mathrm{proj}_v(\mathcal{S}_{\mathrm{safe,k}})$
2: $s_{\min} := \min(\mathrm{proj}_s(\mathcal{S}_{\mathrm{safe,k}}))$
3: `curvedLowerBound` $\leftarrow \emptyset$, `upperBound` $\leftarrow \emptyset$
4: **for** $v_i \in [v_{\min}, v_{\max}]$ **do**
5: $\quad [s_i, s_{i'}] := \mathrm{proj}_{s,v}(\mathcal{S}_{\mathrm{safe,k}})(v = v_i)$
6: $\quad$ Add $(s_{i'}, v_i)$ to `upperBound`.
7: $\quad$ **if** $s_i \neq s_{\min}$ **then**
8: $\quad\quad$ Add (s_i, v_i) to `curvedLowerBound`.
9: $\quad$ **end if**
10: **end for**
11: Sort points in `curvedLowerBound` according to v.
 # Convexify the curved part of lower bound, see Fig. 7
12: $(s_0, v_0) \leftarrow$ first point of `curvedLowerBound`
13: $(s_n, v_n) \leftarrow$ first point of `curvedLowerBound`
14: `line1` $\leftarrow$ computeStraightLine$((s_0, v_0), (s_n, v_n))$
15: `pt` $\leftarrow$ findFurthestPoint(`curvedLowerBound`, `line1`)
16: $\langle A_{\mathrm{iss}}, b_{\mathrm{iss}} \rangle_{\tilde{\mathcal{P}}} \leftarrow$ computeParallelLine(`line1`, `pt`)
 # Add constraints for upper bound, left, and right edge
17: Add `upperBound` to $\langle A_{\mathrm{iss}}, b_{\mathrm{iss}} \rangle_{\tilde{\mathcal{P}}}$.
18: Add $v \geq v_{\min}$ and $v \leq v_{\max}$ to $\langle A_{\mathrm{iss}}, b_{\mathrm{iss}} \rangle_{\tilde{\mathcal{P}}}$.
19: **return** $\langle A_{\mathrm{iss}}, b_{\mathrm{iss}} \rangle_{\tilde{\mathcal{P}}}$

A detailed implementation of convexifying the safe distance set is given in Algorithm 3. In Line 1-11, we extract the waypoints of the upper bound and the curved part of the lower bound. Next, we compute a straight line to inner-approximate the curved part of the lower bound in Line 12-16, where computeStraightLine() computes a straight line

[4]https://sjnavarro.files.wordpress.com/2011/08/aashto-2001.pdf, Exhibit 2-2

TABLE VIII: Observation space definition, taken from [75, Tab. II].

Ego-Vehicle-Related	**Surrounding-Traffic-Participants-Related**
velocity $[\mathrm{m/s}]$	is collision [-]
acceleration $[\mathrm{m/s^2}]$	**Lane-based surrounding detection**
jerk $[\mathrm{m/s^3}]$	distance to surrounding participants $[\mathrm{m}]$
curvature $[\mathrm{/m}]$	relative velocity to surrounding participants $[\mathrm{m/s}]$
derivative of curvature $[\mathrm{/m \cdot s}]$	
Goal-Related	**Road-Network-Related**
Euclidean goal distance $[\mathrm{m}]$	relative offset to center $[\mathrm{m}]$
longitudinal goal distance $[\mathrm{m}]$	left marker distance $[\mathrm{m}]$
lateral goal distance $[\mathrm{m}]$	right marker distance $[\mathrm{m}]$
remaining time steps [-]	left road edge distance $[\mathrm{m}]$
is goal reached [-]	right road edge distance $[\mathrm{m}]$
is time out [-]	is offroad [-]

passing through two points, $\mathrm{findFurthestPoint}()$ returns a point of a given curve that is furtherst to a given straight line, and $\mathrm{computeParallelLine}()$ returns a line that is parallel to a given line and passing through a given point.

D. Observations

We list the used observations in Table VIII, which are adapted from [75, Tab. II].

ACKNOWLEDGMENT

The authors thank Gerald Würsching and Daniel Tar for their help with the numerical experiments.

REFERENCES

[1] A. D. Beza and M. M. Zefreh, "Potential effects of automated vehicles on road transportation: a literature review," *Transp. Telecommun.*, vol. 20, no. 3, pp. 269–278, 2019.

[2] D. González, J. Pérez, V. Milanés *et al.*, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transport. Syst.*, vol. 17, no. 4, pp. 1135–1145, 2015.

[3] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 2765–2771.

[4] B. R. Kiran, I. Sobh, V. Talpaert *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transport. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2021.

[5] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] D. Silver, J. Schrittwieser, K. Simonyan *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[8] S. Gu, L. Yang, Y. Du *et al.*, "A review of safe reinforcement learning: Methods, theory and applications," *arXiv preprint arXiv:2205.10330*, 2022.

[9] C. Pek, S. Manzinger, M. Koschi *et al.*, "Using online verification to prevent autonomous vehicles from causing accidents," *Nat. Mach. Intell.*, vol. 2, no. 9, pp. 518–528, 2020.

[10] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, 2014.

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

[12] N. Mehdipour, M. Althoff, R. D. Tebbens *et al.*, "Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges," *Automatica*, vol. 152, p. 110692, 2023.

[13] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015.

[14] L. Brunke, M. Greeff, A. W. Hall *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 5, pp. 411–444, 2022.

[15] H. Krasowski, J. Thumm, M. Müller *et al.*, "Provably safe reinforcement learning: A theoretical and experimental comparison," *arXiv preprint arXiv:2205.06750*, 2022.

[16] W. Zhao, T. He, R. Chen *et al.*, "State-wise safe reinforcement learning: A survey," in *Int. Joint Conf. Artif. Intell.*, 2023.

[17] B. Mirchevska, C. Pek, M. Werling *et al.*, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2156–2162.

[18] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proc. of the AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[19] D. Liu, M. Brännstrom, A. Backhouse *et al.*, "Learning faster to perform autonomous lane changes by constructing maneuvers from shielded semantic actions," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2019, pp. 1838–1844.

[20] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *Proc. of the IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–6.

[21] M. Bouton, A. Nakhaei, K. Fujimura *et al.*, "Safe reinforcement learning with scene decomposition for navigating complex urban environments," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 1469–1476.

[22] M. Bouton, J. Karlsson, A. Nakhaei *et al.*, "Reinforcement learning with probabilistic guarantees for autonomous driving," in *Workshop on Safety Risk and Uncertainty in Reinforcement Learning*, 2018.

[23] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.

[24] L. M. Schmidt, G. Kontes, A. Plinge *et al.*, "Can you trust your autonomous car? Interpretable and verifiably safe reinforcement learning," in *Proc. of the IEEE Intell. Veh. Symp.*, 2021, pp. 171–178.

[25] K. Mokhtari and A. R. Wagner, "Safe deep Q-Network for autonomous vehicles at unsignalized intersection," *arXiv preprint arXiv:2106.04561*, 2021.

[26] J. Fischer, C. Eyberg, M. Werling *et al.*, "Sampling-based inverse reinforcement learning algorithms with safety constraints," in *Proc. of the IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 791–798.

[27] H. Krasowski, Y. Zhang, and M. Althoff, "Safe reinforcement learning for urban driving using invariably safe braking sets," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2407–2414.

[28] D. Kamran, T. Engelgeh, M. Busch *et al.*, "Minimizing safety interference for safe and comfortable automated driving with distributional reinforcement learning," in *Proc. of the IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1236–1243.

[29] J. Xu, X. Pei, and K. Lv, "Decision-making for complex scenario using safe reinforcement learning," in *Proc. of the CAA Int. Conf. Veh. Control Intell.*, 2020, pp. 1–6.

[30] D. Chen, L. Jiang, Y. Wang *et al.*, "Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model," in *Proc. of the Am. Control Conf.*, 2020, pp. 4355–4361.

[31] C. Zhang, K. Kacem, G. Hinz *et al.*, "Safe and rule-aware deep reinforcement learning for autonomous driving at intersections," in *Prof. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2708–2715.

[32] L. Wen, J. Duan, S. E. Li *et al.*, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–7.

[33] S. Gu, G. Chen, L. Zhang *et al.*, "Constrained reinforcement learning for vehicle motion planning with topological reachability analysis," *Robotics*, vol. 11, no. 4, p. 81, 2022.

[34] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[35] J. Rong and N. Luan, "Safe reinforcement learning with policy-guided planning for autonomous driving," in *Proc. of the IEEE Int. Conf. Mechatron. Autom.*, 2020, pp. 320–326.

[36] Z. Cao, S. Xu, X. Jiao *et al.*, "Trustworthy safety improvement for autonomous driving using reinforcement learning," *Transp. Res. Part C Emerg. Technol.*, vol. 138, p. 103656, 2022.

[37] D. Kamran, Y. Ren, and M. Lauer, "High-level decisions from a safe maneuver catalog with reinforcement learning for safe and cooperative automated merging," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2021, pp. 804–811.

[38] J. Bernhard, S. Pollok, and A. Knoll, "Addressing inherent uncertainty: Risk-sensitive behavior generation for automated driving using distributional reinforcement learning," in *Proc. of the IEEE Intell. Veh. Symp.*, 2019, pp. 2148–2155.
[39] D. Kamran, C. F. Lopez, M. Lauer *et al.*, "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 1205–1212.
[40] X. Xiong, J. Wang, F. Zhang *et al.*, "Combining deep reinforcement learning and safety based control for autonomous driving," *arXiv preprint arXiv:1612.00147*, 2016.
[41] Z. Li, U. Kalabić, and T. Chu, "Safe reinforcement learning: Learning with supervision using a constraint-admissible set," in *Proc. of the Am. Control Conf.*, 2018, pp. 6390–6395.
[42] N. Kochdumper, H. Krasowski, X. Wang *et al.*, "Provably safe reinforcement learning via action projection using reachability analysis and polynomial zonotopes," *IEEE Open J. Control Syst.*, vol. 2, pp. 79–92, 2023.
[43] Y. S. Shao, C. Chen, S. Kousik *et al.*, "Reachability-based trajectory safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3663–3670, 2021.
[44] S. Chen, Y. Sun, D. Li *et al.*, "Runtime safety assurance for learning-enabled control of autonomous driving vehicles," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2022, pp. 8978–8984.
[45] J. Achiam, D. Held, A. Tamar *et al.*, "Constrained policy optimization," in *Proc. of the Int. Conf. on Mach. Learn.*, 2017.
[46] T. P. Lillicrap, J. J. Hunt, A. Pritzel *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[47] W. Dabney, G. Ostrovski, D. Silver *et al.*, "Implicit quantile networks for distributional reinforcement learning," in *Proc. of the Int. Conf. Mach. Learn.*, 2018, pp. 1096–1105.
[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[49] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2007.
[50] J. Schulman, F. Wolski, P. Dhariwal *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[51] W. Dabney, M. Rowland, M. Bellemare *et al.*, "Distributional reinforcement learning with quantile regression," in *Proc. of the AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.
[52] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.
[53] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. of the Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[54] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 249–265, 2020.
[55] C. Pek and M. Althoff, "Efficient computation of invariably safe states for motion planning of self-driving vehicles," in *Proc. of the IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3523 – 3530.
[56] ——, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 798–814, 2020.
[57] B. Schürmann, D. Heß, J. Eilbrecht *et al.*, "Ensuring drivability of planned motions using formal methods," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2017, pp. 1–8.
[58] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction (2nd ed.)*. MIT press, 2018.
[59] J. Schulman, P. Moritz, S. Levine *et al.*, "High-dimensional continuous control using generalized advantage estimation," in *Proc. of the Int. Conf. Learn. Represent.*, 2016.
[60] J. Schulman, S. Levine, P. Abbeel *et al.*, "Trust region policy optimization," in *Proc. of the Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[61] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.
[62] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[63] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," in *IEEE Intell. Veh. Symp.*, 2017, pp. 1686–1693.
[64] E. Irani Liu, G. Würsching, M. Klischat *et al.*, "CommonRoad-Reach: A toolbox for reachability analysis of automated vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2022, pp. 2313–2320.
[65] M. Althoff and S. Magdici, "Set-based prediction of traffic participants on arbitrary road networks," *IEEE Trans. Intell. Veh.*, vol. 1, no. 2, pp. 187–202, 2016.
[66] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 232–248, 2020.
[67] G. Würsching and M. Althoff, "Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2021, pp. 828–835.
[68] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transport. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2017.
[69] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 4, pp. 369–395, 2021.
[70] C. Pek, V. Rusinov, S. Manzinger *et al.*, "CommonRoad drivability checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 1013–1020.
[71] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. of the IEEE Intell. Veh. Symp.*, 2010, pp. 518–522.
[72] E. Velenis and P. Tsiotras, "Optimal velocity profile generation for given acceleration limits; the half-car model case," in *Proc. of the IEEE Int. Symp. Ind. Electron.*, 2005, pp. 361–366.
[73] A. Rizaldi, F. Immler, and M. Althoff, "A formally verified checker of the safe distance traffic rules for autonomous vehicles," in *NASA Form. Methods Symp.*, 2016, pp. 175–190.
[74] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Int. Workshop Alg. Found. Robot.*, 2015, pp. 109–124.
[75] X. Wang, H. Krasowski, and M. Althoff, "CommonRoad-RL: A configurable reinforcement learning environment for motion planning of autonomous vehicles," in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2021, pp. 466–472.
[76] R. Krajewski, J. Bock, L. Kloeker *et al.*, "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 Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2118–2125.
[77] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 719 – 726.
[78] M. Althoff and R. Lösch, "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" in *Proc. of the IEEE Int. Conf. Intell. Transp. Syst.*, 2016, pp. 485–491.
[79] B. Schürmann, M. Klischat, N. Kochdumper *et al.*, "Formal safety net control using backward reachability analysis," *IEEE Trans. Autom. Control*, vol. 67, no. 11, pp. 5698–5713, 2021.

Xiao Wang received the B.Eng. degree in Vehicle Engineering in 2015 from Tongji University, China, and the M.Sc. degree in Mechanical Engineering in 2018 from Technische Universität München, Germany. She is currently a Ph.D. student at Technische Universität München, Germany. Her research focuses on motion planning for autonomous vehicles, formal methods, and safe reinforcement learning.

Matthias Althoff is an associate professor in computer science at Technische Universität München, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from Technische Universität München, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Technische Universität Ilmenau, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.