

Risk-aware controller for autonomous vehicles using model-based collision prediction and reinforcement learning [☆]

Eduardo Candela ^{*}, Olivier Doustaly, Leandro Parada, Felix Feng, Yiannis Demiris, Panagiotis Angeloudis

Centre for Transport Studies, Department of Civil and Environmental Engineering, Imperial College London, London SW7 2AZ, United Kingdom

ARTICLE INFO

Article history:

Received 16 May 2022

Received in revised form 7 November 2022

Accepted 15 April 2023

Available online 20 April 2023

Keywords:

Autonomous vehicles

Decision making

Risk-aware AI

Reinforcement learning

Model-based prediction

Explainable AI

ABSTRACT

Autonomous Vehicles (AVs) have the potential to save millions of lives and increase the efficiency of transportation services. However, the successful deployment of AVs requires tackling multiple challenges related to modeling and certifying safety. State-of-the-art decision-making methods usually rely on end-to-end learning or imitation learning approaches, which still pose significant safety risks. Hence the necessity of risk-aware AVs that can better predict and handle dangerous situations. Furthermore, current approaches tend to lack explainability due to their reliance on end-to-end Deep Learning, where significant causal relationships are not guaranteed to be learned from data.

This paper introduces a novel risk-aware framework for training AV agents using a bespoke collision prediction model and Reinforcement Learning (RL). The collision prediction model is based on Gaussian Processes and vehicle dynamics, and is used to generate the RL state vector. Using an explicit risk model increases the post-hoc explainability of the AV agent, which is vital for reaching and certifying the high safety levels required for AVs and other safety-sensitive applications. Experimental results obtained with a simulator and state-of-the-art RL algorithms show that the risk-aware RL framework decreases average collision rates by 15%, makes AVs more robust to sudden harsh braking situations, and achieves better performance in both safety and speed when compared to a standard rule-based method (the Intelligent Driver Model). Moreover, the proposed collision prediction model outperforms other models in the literature.

© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Recent advances in the fields of Artificial Intelligence (AI) and Deep Learning (DL) in particular have led to an accelerated pace in the development of Autonomous Vehicle (AV) technologies. AVs possess an enormous potential for saving lives, economic growth, reducing commute times, decreasing the environmental impact of transport, among many others. It is estimated that vehicle accidents will take the life of 1.35 million people per year in the world [1], and cost \$1.8 trillion USD to the global economy [2]. However, the state-of-the-art is still far from successfully and safely deploying AVs on roads. The Society of Automotive Engineers (SAE) has defined a 0-5 scale to measure the level of automation of AV [3].

[☆] This paper is part of the Special Issue: "Risk-aware Autonomous Systems: Theory and Practice".

^{*} Corresponding author.

E-mail address: e.candela-garza19@imperial.ac.uk (E. Candela).

Current commercial vehicles only present level 2 autonomy, meaning the driver has to be ready to take control at all times. Achieving higher SAE autonomy levels requires reliable systems that can be certified to meet necessary safety specifications.

Training and testing AVs in real-life poses significant safety risks, hence the necessity for risk-aware AVs that can better predict and handle dangerous situations. AVs are complex systems that autonomously handle diverse driving tasks such as lane-keeping, merging, or parking. The architecture of an AV consists of modules that, based on their functionality, can be grouped into perception (sensors and object detection), planning (generation and prediction of possible trajectories and behaviors), and decision-making. This paper focuses on the strategic decision-making module, which from now on we just call the “AV”.

The problem of training an agent that learns sequential decision-making in a dynamic environment has been successfully handled by Reinforcement Learning (RL). During the past decade, Deep Learning (DL) has been increasingly together with RL, leading to the new field of Deep Reinforcement Learning (DRL). One of the best-known success stories of DRL is when AlphaGo beat the human world champion in the game Go [4]. Using RL allows achieving methods that can outperform rule-based methods by finding counter-intuitive policies that return higher rewards. A clear example is when AlphaZero significantly outperformed AlphaGo by removing all the data and insights from human expert players [5].

A recent success story of using DRL in driving is GT Sophy, the first AI agent to outrace the world's best drivers of the racing simulation game Gran Turismo [6]. Moreover, the generation of AV decision-making policies with DRL has been studied using a variety of approaches, such as: Value-based methods [7], Actor-Critic methods [8,9], and Imitation Learning methods [10,11]. Beyond AVs, there is an increasing research focus on Safe RL, defined as the set of methods that learn policies in problems that require ensuring performance and safety levels [12]. Nevertheless, there is a clear gap in the literature regarding risk-aware AV decision-making methods that use explicit safety models, which is the problem this paper is addressing.

Another challenge in state-of-the-art AVs is the lack of explainability of popular end-to-end Deep Learning approaches, making it hard or impossible to understand why systems fail [13]. In fact, this lack of explainability is one of the reasons impeding the safety certification and adaptation of AVs [14].

Explainable Artificial Intelligence (XAI) allows human users to better interpret, comprehend and trust the outputs of AI-powered systems. Explainability can help ensure that only meaningful variables infer the output, which is needed to guarantee that models capture truthful causality [15]. The XAI literature distinguishes between models that are interpretable by design and those that can be explained using external techniques. See Arrieta, et al. [15] for a survey of XAI definitions and categories based on the most common uses in the literature. Models that use neural networks are not interpretable by design, therefore, require post-hoc explainability, which “targets models that are not readily interpretable by design by resorting to diverse means to enhance their interpretability” [15]. Feature relevance explanation are methods for post-hoc explainability that “clarify the inner functioning of a model by computing a relevance score for its managed variables” [15].

The contributions of this paper are summarized as follows:

1. *A novel risk-aware RL framework for training AVs.* The risk awareness is achieved with a bespoke collision prediction model based on Gaussian Processes and vehicle dynamics, which is used to efficiently generate features for RL state vector that capture risk in 2D space. The framework addresses the following problem in Safe RL: safety cannot be distinguished from the policy when using rewards or constraints to handle safety.
2. *Design of an RL framework that can outperform rule-based methods.* By better handling the trade-off between safety and performance, our proposed RL framework performs better than a standard rule-based method by attaining higher average safety and speed. This is validated with various experiments using a simulator.
3. *Controlling different decision-making driving functions autonomously and simultaneously.* The generality of the proposed RL framework and the action space allows to simultaneously control these driving functions: adaptive cruise control, merging, lane changing, collision avoidance and emergency braking. Whereas usually each driving function would require a separate controller.
4. *Increasing the explainability of the AV agent.* This is achieved with post-hoc explainability using feature relevance explanation. We feed the black-box DRL models with white-box features generated with the collision prediction model, and then assess the difference in performance with and without collision prediction.

The rest of the paper is organized into the following sections. Section 2 contains an overview of the general concepts that conform this study, namely Safe and Risk-aware RL, AV frameworks, and collision prediction. Section 3 provides a literature review on the most relevant research on decision-making for AVs. Section 4 presents the proposed methodology in detail, starting with a formal problem definition, followed by the collision prediction model and the formulation of the RL problem for training AV agents. Section 5 describes the implementation and results of the experiments conducted. Finally, Section 6 discusses the conclusions of this study and suggests future lines of work.

2. Background

This section provides an overview of the general concepts upon which we build our methodology: AV frameworks, Safe and risk-aware RL, and collision prediction.

2.1. AV frameworks

Standard autonomous driving frameworks encapsulate functionalities in modules [16]. There are multiple vehicle control and motion planning methods, with each presenting particular benefits and drawbacks. The selection of methods depends on the level of autonomy and safety required, the driving task at hand, and the desired approach. Pipelines for modular approaches involve perception, object detection, behavior prediction, planning, and decision-making. A modular system has the advantage of increasing the explainability of the models as each task is broken down into simpler sub-tasks. However, it can suffer from human bias and error propagation [17].

There is an increasing interest in researching and deploying methods that take an end-to-end approach: in which perception and planning are integrated. End-to-end methods make use of Deep Neural Networks (DNNs) to control the vehicle directly from sensory inputs [18]. ALVINN (Autonomous Land and Vehicle in a Neural Network) pioneered this approach by training a neural network to directly output steering angles from inputted camera images [19]. However, the insufficient feedback of uncertainty is a problem in neural network-based perception systems [17]. These methods also have downsides, such as a lack of explainability and difficulty in verifying the robustness of the DNNs, making it complex to understand when and why they will fail [20]. A promising approach for tackling this problem in AVs is the further development of XAI tools, which are widely recognized as essential for the practical deployment of AI models [15].

The main development and testing of AV technologies initially happened on real roads, which even up to this day can be dangerous. There are some recent examples of fatal incidents involving self-driving technologies, such as the ones designed by Uber [21] and Tesla [22]. Therefore, simulation has become crucial for developing AVs due to its capacity to replicate failures in a safe environment, together with increased computing power and significantly lower costs. AV simulators' safety, efficiency, and increased realism have made RL essential for AV training.

2.2. Safe and risk-aware RL

Safe RL is a field that has experienced recent growth in applications that require ensuring system performance and adherence to safety levels and constraints, which is the case in AVs. Safety can be related to the concept of *risk* as the implications of the inherent uncertainty or stochasticity of the environment [12]. García et al. [12] wrote a comprehensive survey on Safe RL, in which they introduce two categories for methods that either modify the optimization criterion or the exploration process.

The optimization criterion category consists of methods that adapt returns to manage risk depending on different criteria. The worst-case criterion aims to maximize the worst-case returns in cases where variability is due to the inherent uncertainty of the system (Heger [23]); or due to uncertainty of the unknown parameters of the Markov Decision Process (Nilim and El Ghaoui [24]). The risk-sensitive criterion tries to balance the return and the risk, which can be done with a linear combination (Markovitz [25], Sato et al. [26]), or using the probability of the occurrence of an error state (Geibel and Wysotzki [27]). In the constrained criterion the expected value of some introduced measures should stay within some given bounds (Moldovan and Abbeel [28]). The problem with handling safety with reward or constraints is that knowledge about safety cannot be distinguished from the policy.

Methods belonging to the exploration process category are concerned with modifying the exploratory actions to avoid risky situations, which can be achieved by incorporating external knowledge, or using risk-directed exploration. External knowledge can be provided by examples from a teacher, which are used to initialize the learning algorithm (Driessens and Džeroski [29]); by deriving a policy from a set of expert demonstrations, which replaces the examples gathered by a random exploration policy (Abbeel et al. [30]); or by querying a teacher during the learning process (Thomaz and Breazeal [31]). In risk-directed exploration, a risk measure estimates the probability of selecting an action during the exploration process (Gehring and Precup [32]).

In risk-aware RL, risk can be estimated in various ways, and can be used to modify the optimization criterion, the exploration criterion, or even both. A common approach is defining risk as return variance, and penalizing it in the optimization criterion (Markovitz [25], Sato et al. [26]). However, using the return variance as a measure of risk poses several limitations, such as underestimating risk, not accounting for the fat tails of the distribution, and not distinguishing between beneficial and prejudicial risk [12]. Risk-directed exploration makes use of a defined risk metric to encourage or discourage the exploration of regions of the environment (Gehring and Precup [32]). On the other hand, risk functions can be estimated by repeatedly visiting error states (Geibel and Wysotzki [27], García and Fernández [33]).

2.3. Collision prediction

Collision prediction is essential for both training and deployment of Autonomous Vehicles. Extensive research has been conducted on vehicle collisions, both qualitative and quantitative. Some examples of qualitative studies on driver behavioral factors can be found in [34,35]. Regarding quantitative research on vehicle accidents, the lack of real collision data poses a challenge, which leads to the common use of near-collision or near-miss data. For instance, Wang et al. trained a tree-based machine learning model with near-crash data to do driving risk assessment [36], and Åsljung et al. used Extreme Value Theory to estimate the frequency of collisions with near-collision data [37]. Nonetheless, these approaches are limited by their dependence on near-crash data and their arbitrary definition.

Collision prediction methods can be classified as either learning-based or model-based. Learning-based methods learn a model by finding the parameters that minimize a loss function evaluated on a training data set; most popular methods use Deep Learning. For some examples of research on collision prediction using neural networks: Strickland et al. implemented a Deep Predictive Model with a Bayesian Convolutional Long Short-Term Memory neural network (LSTM) [38]; Chen et al. use V2I (Vehicle-to-Infrastructure) and V2V (Vehicle-to-Vehicle) communication together with a GA-optimized neural network for rear-end collision avoidance [39]; Lee and Yeo predict severe deceleration using a Multilayer Perceptron (MLP) neural network [40]. These methods are highly dependent on the quality and size of training data, making them prone to overfitting. Moreover, neural networks lack explainability, can be difficult to train, and usually converge to local optima.

Model-based methods rely on a priori knowledge of the agents and their environment, usually in the form of dynamics or probabilistic models. One of most simple and commonly used models is Time to collision (TTC), defined as the length of time until a collision if the vehicle continues its current path [41]. Other metrics include Time-integrated time to collision (TIT), Time-exposed time to collision (TET), Post-encroachment time (PET), and Deceleration rate to avoid collision (DRAC).

More complex models can be found in the literature. For instance, Althoff et al. estimate stochastic occupancy of the road using reachable sets for collision prediction [42], and Lambert et al. use Monte Carlo numerical methods to approximate the integral of the product of probability distributions that describes a collision [43]. Both these approaches involve complex calculations that lead to higher computation times.

Trajectory prediction is essential for collision prediction, and the existing approaches can also be classified as learning-based or model-based. Most of the recent literature focuses developing learning-based methods, usually by making use of neural networks that capture temporality. For instance, Strohecker et al. [44] utilize a Temporal Convolutional Network (TCN) with rasterized image and vehicle state information as input; He et al. [45] use an architecture with MLP and LSTM to develop a unifying framework to predict trajectories of different road agents. Model-based methods include Kalman filters, Monte Carlo sampling, hidden Markov models, Bayesian networks, and Multiple-model (MM) methods [46].

The method proposed in this paper uses the model presented by Candela et al. [47], which makes use of discretized Gaussian Processes for future vehicle position estimation. This method is selected for its explainability due to being model-based (vehicle dynamics and Gaussian processes), its fast computation that allows fast training of RL algorithms, and its easy calibration using uncertainty parameters.

3. Related work

AV agents have different abstraction levels and are generally tailored to the specific task under study. They can be classified into three levels of abstraction [48]:

- **Dynamics and Control Level:** lowest level of abstraction – requires the knowledge of the vehicle's detailed model.
- **Tactical level:** focuses on guidance algorithms and cooperation schemes.
- **Strategic level:** algorithms abstract the vehicle dynamics and focus on high-level decision making.

This study focuses on decision-making at a strategic level, where perception is assumed to be a given input. Multiple studies have been conducted that focus on training AV decision-making methods for complex driving tasks. We provide a summary of relevant publications in Table 1, which we categorize using the following characteristics:

- **Rule-based and learning-based.** Rule-based methods include feedback control theory, optimal control, and heuristic algorithms [49]. These are conceptually correct but cannot adapt to unforeseen conditions and deal with uncertainties. Also, manually defining a complete set of rules is challenging [17]. Learning-based methods avoid writing hard-coded rules for specific situations and instead learn rules and parameters from data. Learning-based methods such as ML, DL, and RL are increasingly being used [17].
- **Risk-aware.** Methods that estimate risk to represent the uncertainty of the environment [12].
- **Explicit risk model (sample-free).** Risk can be calculated with an estimation from sampled data or with an explicit risk model.

Rios-Torres and Malikopoulos formulated an optimization framework for coordinating CAVs at merging roadways to achieve smooth traffic flow [49]. Arem et al. presented a cooperative adaptive cruise control (CACC) for CAVs to cooperatively drive [50]. While the use of CAVs is promising and being increasingly researched, the advanced communications systems they require are yet to be deployed and standardized in commercial vehicles and will need shared protocols and further regulation [59]. Chen et al. developed a rule-based gap acceptance algorithm for merging into target lanes [51]. Urmsen et al. used a slot-based method for cooperative merging [52]. Nevertheless, these rule-based approaches are vulnerable to unforeseen situations. Wang and Chan proposed a DRL technique for on-ramp merging that consists of a Long Short-Term Memory (LSTM) architecture used together with a Deep Q-Network (DQN) to approximate the Q-function [7]. Ren et al. presented a method for work zone merging using an off-policy soft actor-critic RL algorithm [8]; their method assumes fully automated, connected, and cooperative vehicles; it also requires the use of a Convolutional Neural Network (CNN). Tram et al. proposed a DRL method with Recurrent Neural Networks (RNN) for navigation of intersections with ve-

Table 1

Summary of relevant literature on AV decision-making.

Category	Author	Approach	Risk-aware	Sample-free	Scenario
Rule-based	Rios-Torres and M. [49]	Hamiltonian analysis	×	×	on-ramp merge
	Arem et al. [50]	stochastic simulation	×	×	on-ramp merge
	Chen et al. [51]	bionic decision-making	×	×	on-ramp merge
	Urmson et al. [52]	slot-based method	×	×	intersection
Learning-based	Wang and Chan [7]	DQN	×	×	on-ramp merge
	Ren et al. [8]	soft actor-critic	×	×	highway work zone merge
	Tram et al. [53]	DQN	×	×	intersection
	Isele et al. [54]	Q-learning	×	×	intersection
	Zhang et al. [55]	soft actor-critic	✓	×	off-road
	Wen et al. [56]	PCPO	✓	×	lane keeping, intersection
	Skoglund [57]	POMDP	✓	×	lane keeping, on-ramp merge
	Nyberg et al. [58]	trajectory planner	✓	×	lane keeping

hicles with changing behaviors [53]. The problem of navigating occluded intersections was studied by Isele et al. using DRL [54].

As discussed in Section 1, DRL methods impede safety certification due to a lack of explainability and interpretability. These methods are also usually limited by complex inputs that lead to longer training periods due to more complex policy evaluation. Additionally, operational issues stemming from the use of DRL approaches are difficult to diagnose and debug for systems developers: sensitivity to hyper-parameters, reward shaping, or complex deep learning models [60]. Increasing the explainability of these methods is therefore critical to validate them, which can be done by reducing the input's size and embedding pre-processed information in the state, such as risk models.

We have conducted a systematic literature search with the proposed keywords and identified only four relevant studies on risk-aware AV decision-making, as shown in Table 1. Zhang et al. integrate the soft actor-critic (SAC) algorithm with control theory-based Lyapunov functions to train a policy introducing safety constraints [55]. Wen et al. propose Parallel Constrained Policy Optimization (PCPO), in which three neural networks are used to approximate the policy function, value function, and an added risk function [56]. Both approaches lack explicit risk models and instead rely on constraints, which can be hard to set and do not necessarily guarantee safety. Nyberg et al. propose a risk measure that captures the likelihood and severity of an AV executing a traffic violation, and then integrates it into a sampling-based trajectory planner [58]. Skoglund modeled the environment as a Partially Observable Markov Decision Process (POMDP) and incorporated a risk function estimated from sampling into the reward function [57]. To estimate risk, both approaches require an accurate probabilistic model to sample from, which can be hard or impossible to estimate in the case of complex AV settings.

According to the systematic literature review, there are only four AV decision-making methods that are learning-based and risk-aware, from which none possess explicit risk models since they estimate risk using sampling. This gap is the one we aim to address with this study.

4. Methodology

This section presents in detail the proposed methodology of this paper, beginning with a formal problem definition. Then, the dynamics-based collision prediction model is formulated. Lastly, the problem setting for the simulated environment used to train the AV with RL is discussed. Namely, the state space, action space, and reward function. A high-level overview of the architecture of the proposed approach is presented in Fig. 1.

4.1. Problem definition

We start by formulating the strategic decision-making problem an AV must address as a Markov Decision Process (MDP). The decisions the AV makes depend on the current state $s_t \in S$ at discrete time-step t , which is a continuous encoding of the environment through agent, other road objects, and road layout). The novel contribution of this paper is using a bespoke collision prediction model as feature in the state space, to make the AV risk-aware. We assume full observability of the state (i.e., the observed states are accurate) and model vehicle dynamics with 2 degrees of freedom (for the longitudinal and lateral motions).

The AV agent has to take a strategic action a^{lon} from the possible longitudinal action space $A^{lon}(s_t)$ at time-step t . An example of a longitudinal action space is $A^{lon}(s_t) = \{\text{soft brake, hard brake, soft accelerate, hard accelerate, do not accelerate}\}$. Similarly, for the lateral motion an action a^{lat} from the possible longitudinal action space $A^{lat}(s_t)$. An example of a longitudinal action space is $A^{lat}(s_t) = \{\text{go straight, change to left lane}\}$. We define a single action space $A(s_t) = A^{lon}(s_t) \cup A^{lat}(s_t)$, and assume only one action can be chosen at each time-step with the purpose of having a single policy that is also simple. When a lateral-motion action is chosen (e.g., change lanes) it must be completed before other lateral-motion actions can be taken.

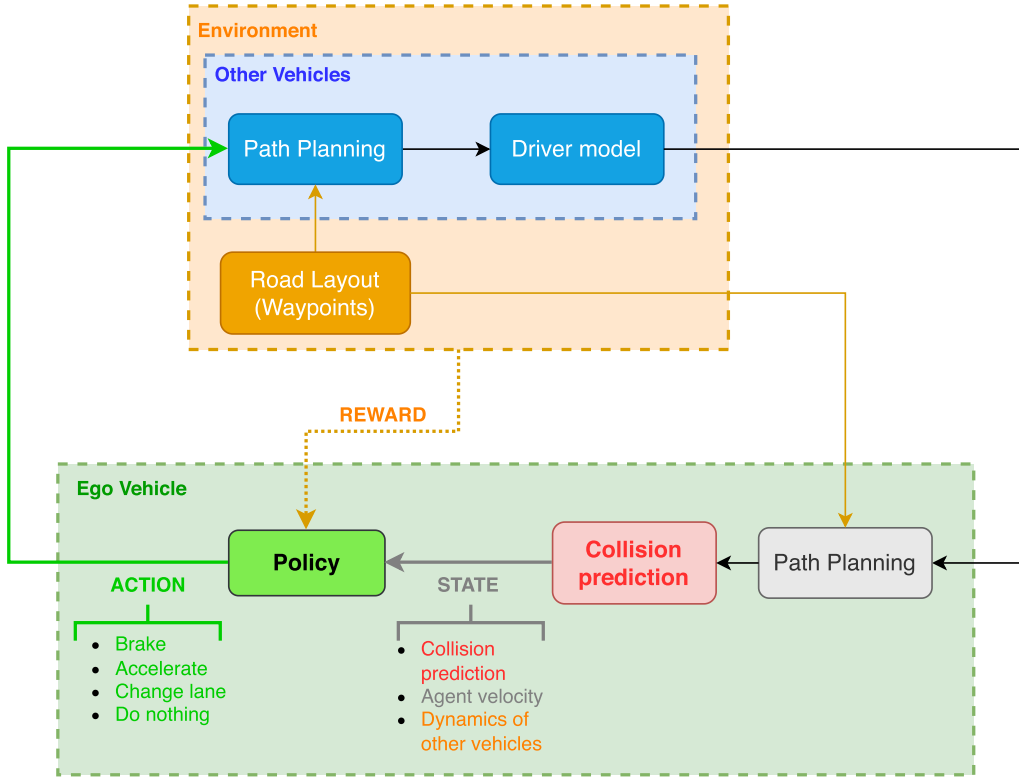


Fig. 1. Architecture of the proposed approach.

The AV agent is represented with a policy $\pi(s|a)$: a function that maps states to actions. To obtain the AV agent policy we use RL, with which policies are evaluated and improved by using the expected future cumulative reward they produce. The value of taking action a in state s under a policy π is denoted by the action-value function or Q-value

$$Q^\pi(s, a) \doteq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right], \quad (1)$$

with discount factor $\gamma \in [0, 1]$ and reward r_t at time t . Therefore, the selected action a_t at time-step t by the policy is the one with the largest Q-value, i.e.,

$$a_t = \pi(s_t) = \operatorname{argmax}_a Q^\pi(s_t, a). \quad (2)$$

An optimal policy π^* is one that corresponds to the optimal action-value function $Q^*(s, a) := \max_\pi Q^\pi(s, a)$ for all $s \in S$ and $a \in A(s)$. Therefore, the goal of RL is finding optimal policy π^* . For more detail on the fundamentals of RL, see Sutton & Barto [61].

Therefore, the input of the policy (the AV agent) is the state vector (which includes the collision prediction), and the output is the chosen strategic action. The diagram in Fig. 1 shows this for the policy module.

4.2. Collision prediction model

The decision-making policy trained in this study uses a vehicle dynamics-based collision prediction model as state feature to simplify the state representation and provide post-hoc explainability of the AV system using feature relevance. The model used is adapted from the one developed by Candela et al. [47], which uses Gaussian Processes for fast collision prediction.

We formulate the collision prediction model as a function that maps vehicle dynamics to a collision prediction grid, given a set of parameters that capture the uncertainty of the dynamics. The function takes as input the predicted dynamics for each vehicle or road object, and the probability of each trajectory, which can be obtained using any trajectory prediction method (see Section 2.3). The parameters that capture the uncertainty of the dynamics are chosen by the user a priori, which can be estimated from data if desired. The output of the function is a grid that contains the probability of a collision happening in 2D space. A high-level diagram of the model is shown in Fig. 2.

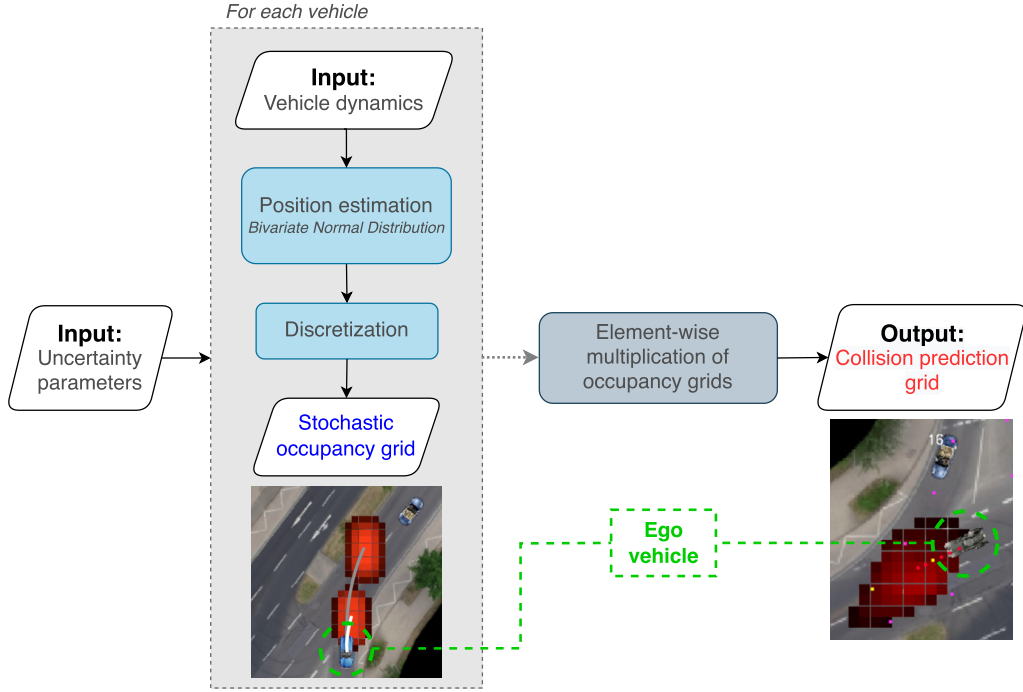


Fig. 2. Diagram of the collision prediction model.

Vehicle dynamics are commonly obtained in discrete time using Euler's method, because reachability algorithms can be challenging due to the interaction between the discrete and continuous stochastic dynamics [62]. Let (x_t, y_t) be the absolute coordinates of the center of a vehicle. We will now derive the discrete dynamics for position x_t as a sum of sequential discrete differences in velocity and acceleration (the logic can be extended to y_t). Let t_0 be the time of the latest known vehicle state, $t > t_0$ the time of the vehicle's future position to be estimated, and $T = \{t_0, t_0 + 1, \dots, t - 1\}$ the set of discrete-time samples. Let f the sampling rate (Hz), $\Delta t = 1/f$ (s), $t_p = t - t_0$ the prediction time s, and $n = ft_p$ (unit-less). We can express the next position as

$$x_{i+1} = x_i + \Delta x_i. \quad (3)$$

We now define $v_i := \Delta x_i$. Note that the unit of v_i is (m), because $v_i \approx v \Delta t$, with instantaneous velocity v . Similarly,

$$v_{i+1} = v_i + \Delta v_i. \quad (4)$$

We define $a_i := \Delta v_i$, which also has (m) as unit because $a_i \approx a(\Delta t)^2$ for instantaneous acceleration a . Therefore, with Euler's method the position at discrete time t is formulated as

$$x_t = x_0 + n v_0 + \sum_{j=0}^{n-1} \sum_{k=0}^{j-1} a_k. \quad (5)$$

To obtain a stochastic position, we start with a deterministic predicted position (x_t^d, y_t^d) for time t . For example, a simple trajectory prediction model would return the position a vehicle is expected to reach when following the same lane at the current speed. To add uncertainty to the deterministic position prediction, we model the stochastic position as a Normal Bivariate random variable:

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} x_t^d \\ y_t^d \end{pmatrix} + \mathbf{E}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \quad (6)$$

where $\boldsymbol{\mu}_t$ is the deterministic position estimation, $\boldsymbol{\Sigma}_t$ the covariance matrix, and $\mathbf{E}_t$ is a random variable capturing the position estimation error. We assume $\mathbf{E}_t$ follows a Normal Bivariate distribution with mean 0 and parameterized variances for the longitudinal and lateral motions that are selected a priori, so that the users can select the positioning error according to their needs. We acknowledge that using Gaussian Processes to estimate stochastic vehicle motion is a simplification of traditional vehicle dynamics. However, the objective of this model is to have a model-based stochastic approach that provides simple and fast prediction of collisions. Fast computation is essential for training policies over thousands of episodes

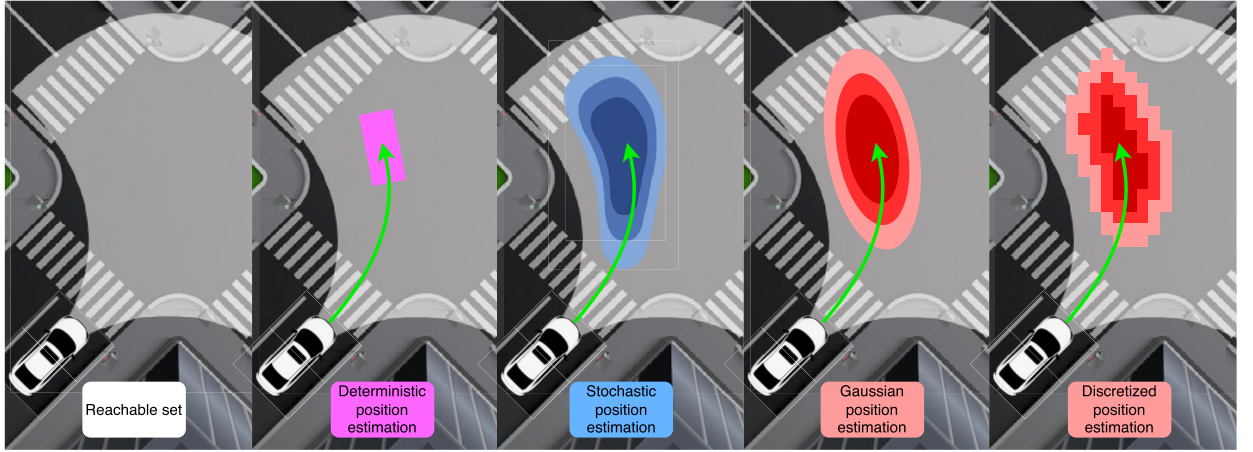


Fig. 3. Position estimation comparison. Our method approximates the stochastic position estimation, which is complex to compute in real time.

using RL, and for deploying in real-time. As discussed in Section 2.3, other collision prediction methods are either Deep Learning-based or computationally expensive to compute, while simpler metrics such as Time to collision (TTC) lack 2D spatial information. See Fig. 3 for a visualization of our method for position estimation, and how it approximates the actual stochastic position, which is hard to compute.

We assume the longitudinal and lateral motions to be independent in order to achieve the required fast computation when training and deploying RL models. Calculating the stochastic position of the vehicle with coupled longitudinal and lateral motions requires significantly more complex computation, as shown in Althoff et al. [42], who also assume the longitudinal and lateral motions to be independent for their probabilistic collision model.

The uncertainty of the position estimation is captured by covariance matrix Σ_t , which is a function of the following parameters selected by the user:

- σ_{lon}^2 : variance in longitudinal acceleration over 1 s
- σ_{lat}^2 : variance in lateral acceleration over 1 s

The larger the prediction time t_p is, the larger the uncertainty of the estimated position. This can be described by replacing in equation (5) the discrete-time accelerations a_k with random variables A_k , which we assume also follow Normal distributions. Note there are $n(n-1)$ terms of discrete accelerations A_k in equation (5). For simplicity, we assume A_{k_1} to be independent of A_{k_2} , $\forall k_1 < k_2$. However, the model can be modified to incorporate the covariances among a sequence of random discrete accelerations, which is beyond the scope of this paper. We can then scale the variance parameters: 1) by $\frac{1}{f^2}$ to adjust it to Δt using sampling rate f , and because $Var(aZ) = a^2 Var(Z)$; 2) by $n(n-1)$ given that $Var(\sum_{i=1}^n Z_i) = \sum_{i=1}^n Var(Z_i)$ when Z_i are all independent. Therefore, the resulting scaling factor is

$$\frac{n(n-1)}{f^2}. \quad (7)$$

We can convert relative vehicle coordinates to absolute coordinates doing a rotation operation. We introduce the rotated relative coordinates (x'_t, y'_t) , such that the vehicle's longitudinal acceleration is parallel to the x'_t axis at each time t , and lateral acceleration is parallel to the y'_t axis. We can then get the absolute coordinates by using rotation matrix $\mathbf{R}_t$, which is computed using heading angle θ_t .

Therefore, the covariance matrix Σ_t in absolute coordinates is obtained by constructing diagonal matrix

$$\mathbf{D}_\tau = \begin{pmatrix} \frac{n(n-1)}{f^2} \sigma_{lon}^2 & 0 \\ 0 & \frac{n(n-1)}{f^2} \sigma_{lat}^2 \end{pmatrix} \quad (8)$$

with the adjusted variance parameters for each time-step $\tau \in T$, and rotating it with $\mathbf{R}_\tau$. Thus, the final covariance matrix

$$\Sigma_t = \sum_{\tau=t_0}^{t-1} \mathbf{R}_\tau^T \mathbf{D}_\tau \mathbf{R}_\tau \quad (9)$$

A position estimation is generated for each object $o \in O$, and each of its predicted trajectories $i \in I^o$ with probability q_i and $\sum_{i \in I^o} q_i = 1$. When the position estimations are calculated, a discrete occupancy grid is generated, which represents the probability of an object occupying grid square $g_{x,y}$ at time t based on the approximation

$$G_{x,y}^{o,t} = \sum_{i \in I^o} q_i \mathbb{P} \left(-\frac{\hat{x}_t^i}{2} \leq x \leq \frac{\hat{x}_t^i}{2} + g_l, -\frac{\hat{y}_t^i}{2} \leq y \leq \frac{\hat{y}_t^i}{2} + g_l \right) \quad (10)$$

where $\hat{x}_t^i = l|\cos(\theta_t^i)| + w|\sin(\theta_t^i)|$, $\hat{y}_t^i = l|\sin(\theta_t^i)| + w|\cos(\theta_t^i)|$, with vehicle length l , width w , and orientation θ_t^i for trajectory i at time t , $x \in M$ corresponds to the grid space along the x -axis, and $y \in N$ along the y -axis. Therefore, $G^{o,t} \in [0, 1]^{M \times N}$. Furthermore, notice that $G_{x,y}^{o,t} = \mathbb{P}(\text{object } o \text{ occupies } g_{x,y} \text{ at time } t)$.

Finally, to predict a collision between objects o_1 and o_2 at time t a new matrix

$$C^{o_1, o_2, t} = G^{o_1, t} \circ G^{o_2, t} \quad (11)$$

is computed with the Hadamard or element-wise product $\circ$ of the two vehicles' stochastic occupancy matrices, where the value of $C_{x,y}^{o_1, o_2, t}$ is the probability of both vehicles occupying the same corresponding grid square $g_{x,y}$. We assume a collision occurs when two or more vehicles occupy the same grid square $g_{x,y}$ at the same time. The pseudo-code for an algorithm of this collision prediction method is shown in Algorithm 1.

Algorithm 1: Collision prediction algorithm.

```

Input:  $\sigma_{lon}^2, \sigma_{lat}^2, g_l, f, n, t_0, t$ ,
Input:  $x_t^{i,o}, y_t^{i,o}, I^o, w^o, q_i, \theta_t^{i,o} \forall$  trajectories  $i \in I^o, \forall$  objects  $o \in O; \sigma_{lon}^2, \sigma_{lat}^2, g_l$ 
Output: Collision prediction grid  $C^t \in [0, 1]^{M \times N}$ 
1  $C^t \leftarrow 0^{M \times N}$ 
2 for object  $o \in O$  do
3   for trajectory  $i \in I^o$  do
4      $\mu_t^{i,o} \leftarrow \begin{pmatrix} x_t^{i,o} \\ y_t^{i,o} \end{pmatrix}$ 
5      $D_\tau \leftarrow \begin{pmatrix} \frac{n(n-1)}{f^2} \sigma_{lon}^2 & 0 \\ 0 & \frac{n(n-1)}{f^2} \sigma_{lat}^2 \end{pmatrix}$ 
6      $\Sigma_t^{i,o} \leftarrow \sum_{\tau=t_0}^{t-1} R_\tau^T D_\tau R_\tau$ 
7      $\hat{x}_t^{i,o} \leftarrow l|\cos(\theta_t^{i,o})| + w|\sin(\theta_t^{i,o})|, \hat{y}_t^{i,o} \leftarrow l|\sin(\theta_t^{i,o})| + w|\cos(\theta_t^{i,o})|$ 
8     /* This can be optimized by doing recursively and ignoring cells with low probabilities */
9      $G_{x,y}^{o,t} \leftarrow \sum_{i \in I^o} q_i \mathbb{P}_{\mathcal{N}(\mu_t^{i,o}, \Sigma_t^{i,o})} \left( -\frac{\hat{x}_t^{i,o}}{2} \leq x \leq \frac{\hat{x}_t^{i,o}}{2} + g_l, -\frac{\hat{y}_t^{i,o}}{2} \leq y \leq \frac{\hat{y}_t^{i,o}}{2} + g_l \right)$ 
10 for object  $o \in O \setminus \{\text{ego}\}$  do
11    $C^t \leftarrow C^t + G^{ego,t} \circ G^{o,t}$ 

```

4.3. Formulation as an MDP for RL

We formulate the problem as an MDP that can be solved by training a policy with RL, with a discrete action space and that works in discrete time. The obtained policy belongs to the Strategic decision-making level, where the vehicle trajectories follow predefined paths and take high-level decisions [48]. For the purposes of this study, we are using a model-free reinforcement learning approach, which avoids the need to explicitly introduce a model with the state-transition probabilities of the environment. Although a probability transition model provides more information about the environment and can allow more accurate estimations, it can be hard to obtain.

It is assumed the agent (ego vehicle) and other vehicles will follow the paths calculated by a path planning module using waypoints from the road layout, and that vehicles do not reverse (i.e., longitudinal velocity must be non-negative). These assumptions are common for decision-making agents that work at a strategic level [48], and the use of a set of possible paths is common in the literature [63,64]. For the other agents, the probabilities of their possible paths are assigned using the Autonomous Multiple Model Estimation (AMM) algorithm [46]. For more detail on the implementation of the path planning module, see Appendix B.

4.3.1. State representation

The state representation is critical to the model as it conditions the agent's perception of the environment. A good state representation is crucial for efficient learning: it should include all the necessary information for decision-making, but it should also be as compact as possible for the agent to learn an optimal policy.

The entries of the state vector are given in Table 2. It encodes the dynamics of the AV agent, the dynamics of the closest vehicles, the immediate road layout (if a lane change or exit is possible with an available connected waypoint), and collision information. Notice that multiple collision predictions are generated for different times in the future, allowing the agent to foresee if a collision is likely to happen given the planned trajectory at various points in the short term. The single collision probabilities are obtained by getting the maximum value over the collision prediction grid returned by the collision prediction model. The three future times {0.6 s, 1.0 s, 1.4 s} for the state vector were chosen to meet the following criteria:

Table 2
State Vector.

Variable	Symbol	Set of values
Agent longitudinal velocity	v	$\mathbb{R}^+$
Distance to front vehicle	d_f	$\mathbb{R}^+$
Distance to rear vehicle	d_r	$\mathbb{R}^+$
Longitudinal velocity of front vehicle	v_f	$\mathbb{R}^+$
Longitudinal velocity of rear vehicle	v_r	$\mathbb{R}^+$
Available waypoint on right side	w_r	$\{0,1\}$
Available waypoint on left side	w_l	$\{0,1\}$
1.4s collision prediction	$c_{1.4}$	$\{0,1\}$
1.0s collision prediction	c_1	$\{0,1\}$
0.6s collision prediction	$c_{0.6}$	$\{0,1\}$
Actual collision	c_0	$\{0,1\}$

- Focus on a short-term planning horizon (2 seconds), to prioritize avoiding immediate collisions. We leave the expansion of our framework to longer planning horizons as future work.
- Keep the average step computing time under 0.1 seconds, to allow deployment at 10 Hz.
- Values that are multiples of 0.2, since the used sampling rate was 5 Hz.
- No consecutive times, to reduce search space.
- The combination of values that yield the safest evaluation results (lowest collision rates) using the proposed methodology. These auxiliary results can be found in Appendix C.

An actual collision c_0 is assumed to happen when there is an overlap of the rectangular surfaces of 2 vehicles or objects. Making use of the full 2D collision prediction grid in the state space is left as future work.

4.3.2. Action

The car is modeled with two degrees of freedom: longitudinal (acceleration and braking) and angular (steering). The steering angle is calculated to point the ego car toward the next way-point in the path generated by the path planning module. This leads to the simplification of the action space, which is discretized to the following actions:

- **Do nothing** – inertia with a deceleration of 0.5 m/s^2 .
- **Soft brake** – negative acceleration of 1.5 m/s^2 .
- **Hard brake** – negative acceleration of 3 m/s^2 .
- **Soft accelerate** – at 2 m/s^2 .
- **Hard accelerate** – at 4 m/s^2 .
- **Left lane change** – when road layout allows, set as target waypoint the next one on the left.
- **Right lane change** – when road layout allows, set as target waypoint the next one on the right.

The simplification of continuous acceleration into a set of a few discrete accelerations has been made to reduce the size of the action space. However, given the frame rate of 5 Hz, different sequences of acceleration, inertia and braking actions enable different compounded acceleration rates over longer periods of time. The use of discrete actions is common in the literature [65,66]. On the other hand, for truly variable acceleration, continuous-action RL models need to be used. Nevertheless, our study focuses on discrete-action models for better understanding of risk-aware policies. We leave the extension to more granular discrete sets and continuous-action RL models as future work.

The braking and acceleration values were selected using as reference the study of Bokare et al. on vehicle acceleration and braking behaviors [67]. While this method proposes a discrete action space for simplicity, it can also be easily adapted to work with a continuous action space, since the collision prediction model and state vector do not require discrete actions.

4.3.3. Reward

The reward function is another critical part of the RL model. When wrongly shaped it can give undesired incentives, leading to inefficient policies, or can make it difficult to achieve convergence. A simple reward function is designed and can be found in Table 3.

It is important to highlight some aspects of the reward function that guarantee a correct logic: collisions are penalized more than the benefit of reaching the final position ($0 < r_{final} < |r_{col}|$); the agent is incentivized to reach the goal as soon as possible ($r_{step} < 0$ and the cumulative penalty is monotonically non-decreasing). We can identify three types of state trajectories or episode endings in the defined problem: a collision ($\phi_{collision}$), not reaching the goal (ϕ_{fail}), reaching the goal (ϕ_{goal}). Clearly, the preferred ordering of these trajectories is $\phi_{collision} < \phi_{fail} < \phi_{goal}$. The cumulative rewards ensure this ordering is respected (when the episode is limited to 2000 steps).

Table 3
Reward function for RL.

Case	Symbol	Reward
Reaching the destination	r_{final}	1000
Collision ($c_0 = 1$)	r_{col}	-2000
Time penalty at each step	r_{step}	-1

Table 4
RL training parameters.

Parameter	Value	Parameter	Value
Learning Rate α	0.001	Initial ϵ (ϵ -greedy exploration)	1
Discount Factor γ	0.99	ϵ decay (per episode)	0.996
Training episodes	2000	min ϵ	0.01
(a) For all RL algorithms.		First layer size	512
		Second and third layer size	256
		Replay memory buffer	500,000
		Batch size	64
		(b) DQN.	
Parameter	Value	Parameter	Value
Learning Rate β	0.001	λ	0.95
Actor first layer size	256	Clip range	0.2
Critic first layer size	256	Network width	64
Actor second layer size	128	Batch size	64
Critic second layer size	128	Entropy coefficient	0.001
(c) A2C.		(d) PPO.	

Simplicity in the reward function allows better interpretability of the results and makes reward design easier [61]. Although the arbitrary selection of the reward function is common in the RL literature, we acknowledge this as a limitation and leave the optimization of the reward function design as future work.

5. Experiments

This section contains a description of the experiments conducted in a simulator to evaluate the performance of the proposed methodology. It starts with the details of the models and parameters used in the implementation of the experiments, followed by an analysis of the results obtained.

5.1. Implementation

5.1.1. RL algorithms

Three state-of-the-art RL algorithms – DQN, A2C, PPO – were implemented to train policies with the proposed methodology. The training parameters used for each RL algorithm are given in Table 4, and a brief description of the algorithms is provided below.

A policy $\pi(a|s)$ is a function that assigns a probability to taking action a given an environment state s . Therefore, a policy can be used as a high-level AV controller. Policies are evaluated by the expected future cumulative reward they produce. The value of taking action a in state s under a policy π is denoted by the action-value function

$$Q^\pi(s, a) \doteq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right], \quad (12)$$

with discount factor $\gamma \in [0, 1]$ and reward r_t at time t . An optimal policy π^* is one that corresponds to the optimal action-value function $Q^*(s, a) := \max_\pi Q^\pi(s, a)$ for all $s \in S$ and $a \in A(s)$. For more detail see Sutton & Barto [61].

As described in subsection 4.3, the tackled AV decision-making problem is framed to have a continuous state space S , and discrete action space A . Therefore, three state-of-the-art RL algorithms that yield policies for continuous state spaces and discrete action spaces were implemented: DQN, A2C, and PPO. Furthermore, they are model-free, meaning they do not require an explicit transition probability distribution. The A2C and PPO methods were implemented to work with a discrete action space as shown by [68].

The Deep Q-Network (DQN) algorithm [69] was developed by enhancing the Q-Learning algorithm with deep neural networks and using an experience replay buffer. A function approximator $Q(s, a; \theta)$ is trained with a neural network with parameters θ to estimate $Q^*(s, a)$. The used loss function at each step i is $L_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim \rho} [(y_i - Q(s, a; \theta_i))^2]$, where $y_i = r_i + \gamma \max_{a'} Q(s', a'; \theta_{i-1})$ is the TD (temporal difference) target, $y_i - Q$ the TD error, and ρ is the behavior distribution (the distribution over transitions $\{s, a, r, s'\}$ gathered from the environment).

Table 5
Collision prediction model parameters.

Variable	Units	Value
Longitudinal acceleration variance σ_{lon}^2	(m/s ²) ²	25
Lateral acceleration variance σ_{lat}^2	(m/s ²) ²	16
Grid length g_i	m	0.25
Sampling rate f	Hz	5
Collision probability ignore threshold	Probability	10e ⁻⁵

The Advantage Actor-Critic (A2C) algorithm [70] combines value-based and policy-based methods. As an Actor-Critic method, A2C consists of two neural networks: an Actor that chooses an action by estimating the optimal policy π^* , and a Critic that evaluates the quality of a given state by estimating the value function $V(s) = \mathbb{E}_{\pi} [R_t | s_t = s]$. However, A2C makes use of an advantage function $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ instead of the value function, improving stability during training. Usually, the advantage function is approximated with an estimator such as the TD error, e.g., $A(s_t, a_t) \approx r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$. The Actor network returns a probability for each action that corresponds to the confidence of that action being the optimal.

Proximal Policy Optimization (PPO) is a policy gradient method introduced by Schulman et al. [71]. PPO aims to simplify the Trust Region Policy Optimization (TRPO) method [72], which maximizes a surrogate objective. This is formulated as $\max_{\theta} L^{CPI}(\theta) = \max_{\theta} \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \hat{A}_t \right]$ subject to the constraint $\hat{\mathbb{E}} [KL[\pi_{\theta_{old}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]] \leq \delta$, where: $\hat{\mathbb{E}}$ is the empirical average, $\hat{A}_t$ is an estimator of the advantage function at time-step t , θ the vector of policy parameters, and θ_{old} the one before the update. Let $r_t(\theta) = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)}$, which is a probability ratio for which $r(\theta_{old}) = 1$. PPO modifies the objective function to penalize changes to the policy that make $r_t(\theta)$ diverge from 1, by using as objective $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t [\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$.

5.1.2. Simulation environment

The method presented in Section 4 was used to obtain AV agents by training a policy for the ego vehicle (the agent) on a bespoke simulator. The simulator was developed to incorporate the collision prediction model, and to achieve faster frame rates by avoiding the use of highly-detailed and complex simulators that were unnecessary for this study. For a systematic review on AV simulators, we refer the reader to [73].

Each scenario consisted of an ego vehicle with an initial and final position, and a path between them defined by the path planning module using waypoints. The agent's objective, as reflected by the reward function shown in Table 3, was to reach a final destination in the shortest amount of time whilst avoiding colliding with other objects.

The other objects included cars, trucks, and bicycles that were simulated with an implementation of the commonly used [74] Intelligent Driver Model (IDM) [75] and Gipps' car following model [76]. While the motion of the objects was controlled with the IDM-Gipps model, the initial configuration of the scenarios needed to be selected. To have more realistic simulated scenarios, we randomly sampled the initial configuration of the scenarios from the open-source data set inD [77], which contains naturalistic trajectories of various types of road agents using drones. From the inD dataset, we randomly sampled: the road layout (highway, intersection, merging ramp, etc.), the start and goal points of the road objects, their properties (type and dimensions), and the traffic density (number of spawned objects). For more detail on the specific implementation of the driver model, see Appendix A; for the path planning module, Appendix B.

A speed limit of 20 m/s (72 km/h) was imposed for the agent and all the other road objects. Furthermore, some scenarios introduced harsh sudden braking by the other vehicles to test the performance of the proposed method in more dangerous scenarios. An illustration of the simulation environment can be found in Fig. 4.

A sampling rate of 5 Hz was used throughout the simulations, which were run on the bespoke simulator developed in Python, on a workstation with an AMD Ryzen Threadripper 3990 64-core processor, 128 GB of RAM, and an Nvidia RTX 2070 GPU.

Regarding the collision prediction model used in the state representation of the environment, the model parameters shown in Table 5 were selected as suggested by [47]. They follow the study of [67] and set the variance parameters such that most of the probability mass ($\sim 95\%$) covers the required value ranges for each estimation. The threshold used for ignoring low probabilities is $10e^{-5}$, which is a lower bound on the estimated probability of a human driver being involved in a collision during the required time frame [78].

5.1.3. List of experiments

A series of experiments were conducted to evaluate the performance of the proposed risk-aware RL method for AV agents. To understand the impact of the risk awareness provided by the collision prediction model, the same method was implemented but *without the collision prediction model* as feature of the state vector. This comparison provides feature relevance post-hoc explainability to the proposed model [15]. Three state-of-the-art RL algorithms were used for training the policies: DQN, A2C, and PPO. To analyze the benefit in performance provided by RL over rule-based methods, the IDM-Gipps model was implemented to control the ego vehicle (the same model used for simulating the other vehicles, see Appendix A).

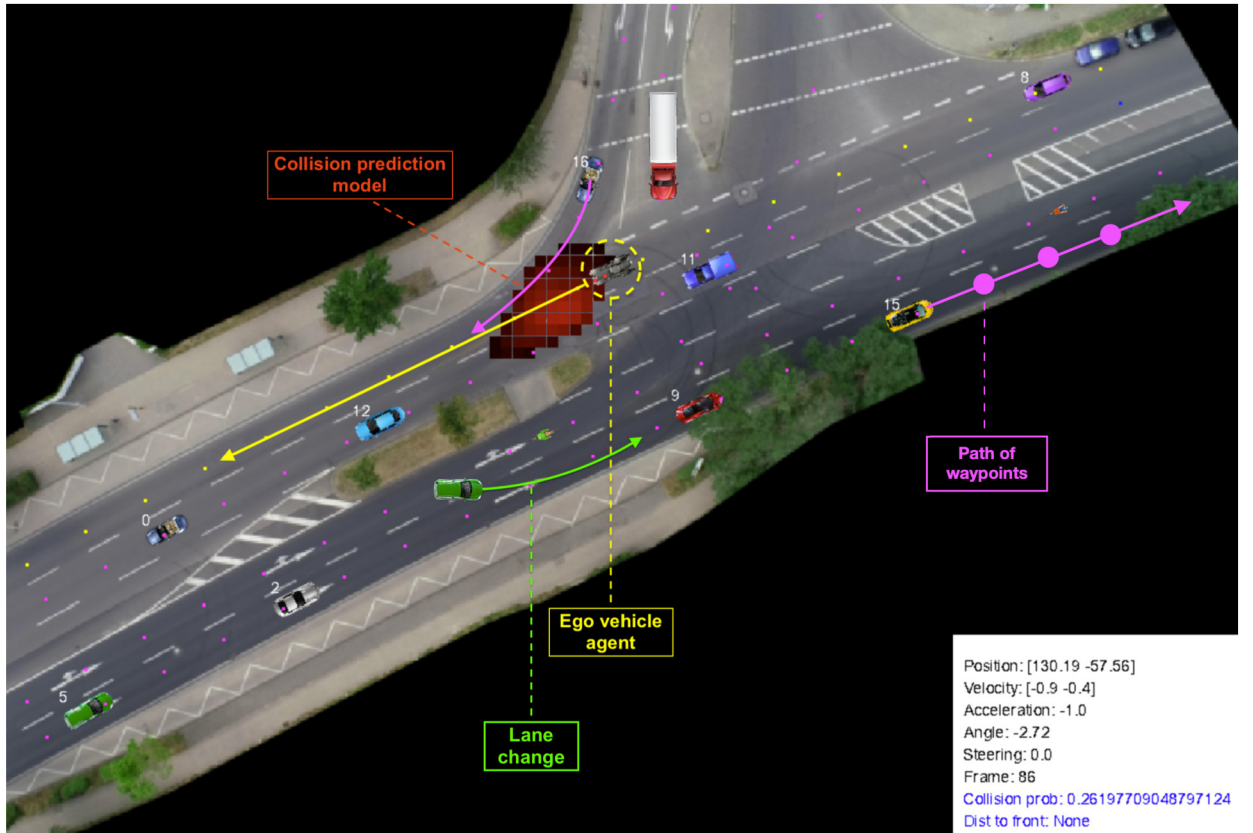


Fig. 4. Highway simulation environment. The ego vehicle is trained to reach the final position following a path of waypoints. The goal is to do so in the shortest amount of time, while avoiding colliding with other vehicles and bicycles.

Table 6

List of experiments: Risk awareness.

Experiment name	Decision-making model	RL algorithm	Sudden braking
Risk-aware DQN no brake DQN no brake	Risk-aware RL policy RL policy without collision prediction	DQN	No
Risk-aware A2C no brake A2C no brake	Risk-aware RL policy RL policy without collision prediction	A2C	
Risk-aware PPO no brake PPO no brake	Risk-aware RL policy RL policy without collision prediction	PPO	
Rule-based no brake	Rule-based IDM-Gipps	–	
Risk-aware DQN brake DQN brake	Risk-aware RL policy RL policy without collision prediction	DQN	Yes
Risk-aware A2C brake A2C brake	Risk-aware RL policy RL policy without collision prediction	A2C	
Risk-aware PPO brake PPO brake	Risk-aware RL policy RL policy without collision prediction	PPO	
Rule-based brake	Rule-based IDM-Gipps	–	

Additionally, another set experiments were run to evaluate how robust the methods are when faced with harsh sudden braking by the other vehicles. A harsh sudden braking event was characterized by a vehicle suddenly braking at 5 m/s^2 until a complete stop (instead of the normal braking acceleration of 3 m/s^2), and was programmed to happen with a probability of 0.05 for each simulated scenario.

A list with the names and parameters of the experiments is shown in Table 6.

Additionally, to benchmark the performance of the proposed collision prediction model (Gaussian), we replaced it in our methodology with other commonly used collision prediction models: Time to collision (TTC), and an implementation of the

Table 7

List of experiments: Collision prediction model comparison.

Experiment name	Collision prediction model	RL algorithm	Sudden braking
Gaussian DQN no brake Althoff DQN no brake TTC DQN no brake	Gaussian Althoff TTC	DQN	No
Gaussian A2C no brake Althoff A2C no brake TTC A2C no brake	Gaussian Althoff TTC	A2C	
Gaussian PPO no brake Althoff PPO no brake TTC PPO no brake	Gaussian Althoff TTC	PPO	
Gaussian DQN brake Althoff DQN brake TTC DQN brake	Gaussian Althoff TTC	DQN	
Gaussian A2C brake Althoff A2C brake TTC A2C brake	Gaussian Althoff TTC	A2C	
Gaussian PPO brake Althoff PPO brake TTC PPO brake	Gaussian Althoff TTC	PPO	

model developed by Althoff et al. [42] (using the parameters they suggest and the same future time values {0.6 s, 1.0 s, 1.4 s}); for more details see Section 2.3. The list of these additional experiments is in Table 7.

For each experiment the same 5 random seeds were used. The RL policies were trained for 2000 episodes with the parameters in Table 4, and all the policies (including rule-based IDM-Gipps) were evaluated on 1000 episodes.

5.2. Results

5.2.1. Visualization of results

The policies were trained and evaluated on various scenarios, where road layouts were sampled from the inD dataset [77]. Depending on the layout of the scenario and the random goal point, the ego agent had to handle different tasks, such as merging onto a highway, changing lanes, crossing an intersection, turning to another street, and emergency braking. A visualization of some of these scenarios can be seen in Fig. 5. Given a situation, which is encoded by the state, the AV agent takes an action using the trained policy. A numerical example of how the policy works is provided in Fig. 6.

5.2.2. Training

All policies were trained during 2000 training episodes using different 5 random seeds. The average rolling rewards during training for the risk awareness experiments (risk-aware RL vs. RL vs. rule-based; see Section 5.1.3) are shown in Fig. 7. Notice that all the risk-aware policies (the ones trained with the methodology of RL with the collision prediction model) converge to higher reward values than their RL counterparts without the collision prediction model. While all the PPO policies achieve better results than the policies trained with DQN and A2C, the risk-aware PPO policy behaves slightly better than its counterpart – a more detailed comparison in performance is presented next in the Evaluation analysis.

Although the rule-based IDM-Gipps model does not require training, it is shown for comparative purposes. The risk-aware RL methodology achieves higher reward during training than the IDM-Gipps model in the following cases: A2C without sudden harsh braking, PPO without sudden braking, and PPO with sudden braking.

For the experiments with RL with the different collision prediction models (proposed Gaussian model vs. Althoff's model vs. TTC; see Section 5.1.3), the average rolling rewards during training are shown in Fig. 8. Notice that during training with DQN and A2C, the policies trained with the proposed Gaussian model converge to higher values, followed by Althoff's model, and then TTC. When training with PPO in the experiments the three models behave similarly during training, except in the environment with sudden harsh braking, where TTC seems to perform worse. A more thorough performance analysis is presented next.

5.2.3. Evaluation

Each of the trained policies was evaluated on 1000 episodes with 5 random seeds. The average rolling rewards during evaluation for the risk awareness experiments (risk-aware RL vs. RL vs. rule-based; see Section 5.1.3) are shown in Fig. 9. All the risk-aware policies achieve higher rewards than their RL counterparts without the collision prediction model. Moreover, the RL risk-aware policies outperform rule-based IDM-Gipps model with A2C in the environment without sudden harsh braking, and with PPO in both environments with and without sudden braking.

Regarding the comparison among the different collision prediction models used as features for the RL state vector, the proposed Gaussian model outperforms Althoff's model and TTC across all experiments, as seen in the plots in Fig. 10.

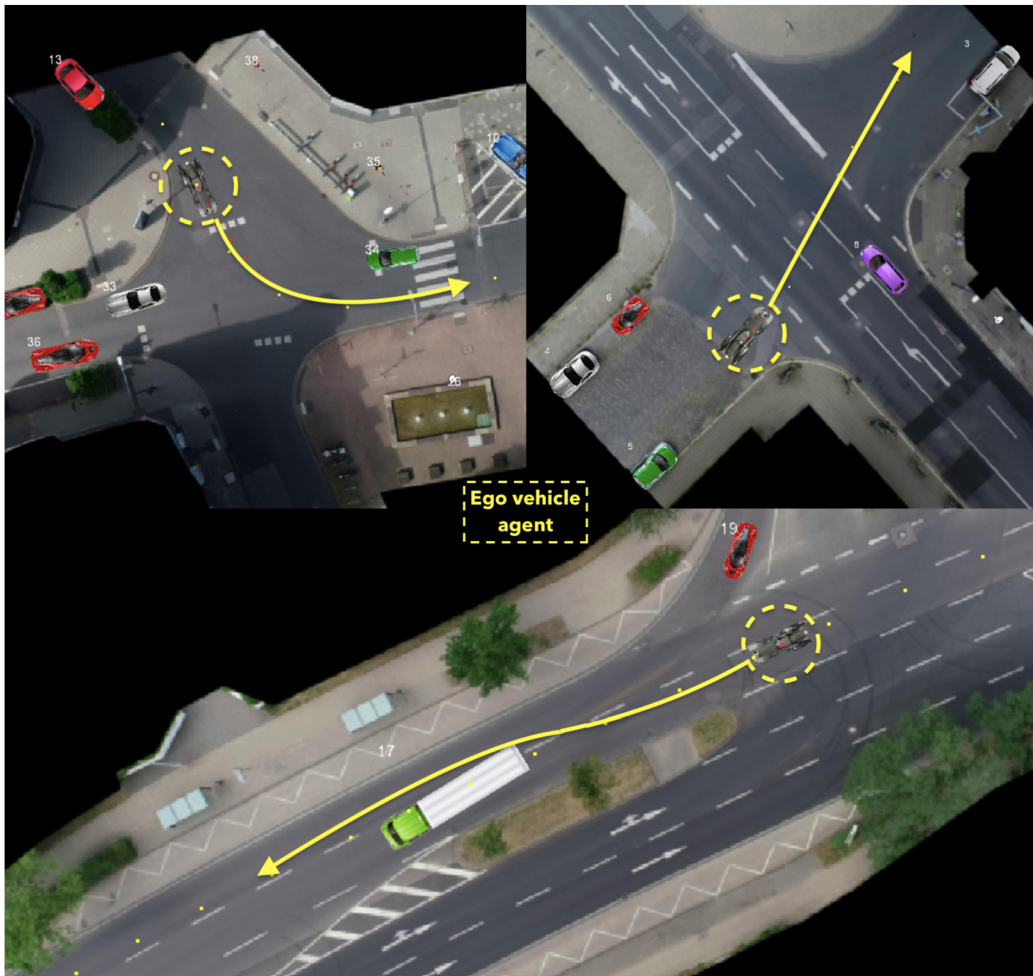


Fig. 5. Examples of simulated scenarios. Top left: left turn; Top right: intersection crossing; Bottom: lane changing.

It is clear that the RL algorithm used for training has a significant impact in the rewards achieved by the policies, with PPO proving to be better in general.

We also analyze the improvement in safety obtained with the proposed methodology, which we measure with the average collision rate achieved by each policy during evaluation. The collision rates for the risk awareness experiments are found in Table 8. Note that the use of the collision prediction model causes a significant reduction in collision rates across all RL algorithms and also with or without sudden harsh braking. The most dramatic improvement of 73% happens when including the risk awareness in scenarios without sudden braking and training with A2C (Table 8a). Moreover, PPO in general outperforms rule-based IDM-Gipps in collision rate as well, with the best result achieved by risk-aware PPO. In fact, risk-aware PPO achieves a collision rate 2.5 times smaller than IDM-Gipps without sudden braking, and 4.0 times smaller with sudden braking. On the other hand, the collision rates remain almost the same with DQN and PPO in the presence of sudden harsh braking, which shows the robustness and adaptability that RL methods can achieve in different scenarios.

The collision rates for the comparison among collision prediction models with RL is in Table 9. The policies trained with the proposed Gaussian collision prediction model outperform the ones trained with Althoff's model and TTC. Althoff's model gives better results than TTC when using A2C, but the opposite happens with PPO. Once again, training with PPO yields the best overall results.

To better understand the behavior of the different trained policies, we analyze the distributions of the agent vehicle's speed and following distance obtained by the different policies.

The distributions for the risk-aware experiments are shown with box-plots in Figs. 11 and 12, for the cases without and with sudden harsh braking, respectively. Similarly, the results for the experiments comparing the use of the different collision prediction models with RL are shown in Figs. 13 and 14.

Finally, we compare the run-time of the different policies, which is an important factor for real-time deployment. The average time it took to compute a step in the simulator with each policy is listed in Table 10. The use of RL models increases computing time, when compared to rule-based IDM-Gipps. Although the Gaussian model takes longer to compute, it is still

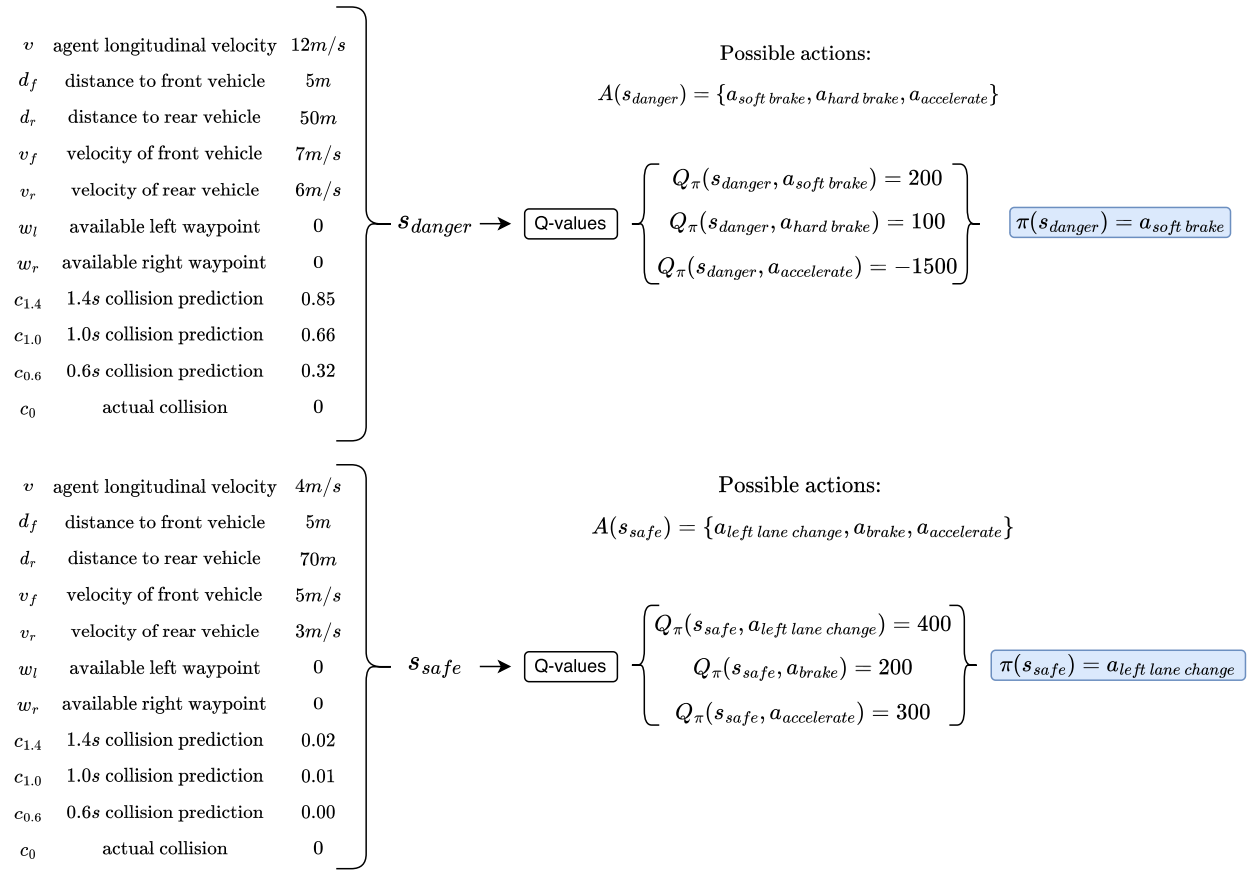


Fig. 6. Examples of policy behavior in different states. In the dangerous state, with likely predicted collisions, the policy chooses to soft brake since that action returns the highest Q-value (expected future cumulative reward). In the safe state, changing lanes yield a higher Q-value and thus is that action selected.

Table 8

Collision rates during evaluation (risk-aware experiments).

Policy	Collision rate
Risk-aware DQN	0.105
DQN	0.132
Risk-aware A2C	0.015
A2C	0.748
Risk-aware PPO	0.008
PPO	0.018
Rule-based IDM-Gipps	0.020

(a) Without sudden harsh braking.

Policy	Collision rate
Risk-aware DQN	0.100
DQN	0.189
Risk-aware A2C	0.146
A2C	0.166
Risk-aware PPO	0.009
PPO	0.020
Rule-based IDM-Gipps	0.037

(b) With sudden harsh braking.

Table 9

Collision rates during evaluation (collision prediction model comparison).

Policy	Collision rate
Gaussian DQN	0.105
Althoff DQN	0.215
TTC DQN	0.158
Gaussian A2C	0.015
Althoff A2C	0.109
TTC A2C	0.683
Gaussian PPO	0.008
Althoff PPO	0.031
TTC PPO	0.025

(a) Without sudden harsh braking.

Policy	Collision rate
Gaussian DQN	0.100
Althoff DQN	0.136
TTC DQN	0.148
Gaussian A2C	0.146
Althoff A2C	0.153
TTC A2C	0.166
Gaussian PPO	0.009
Althoff PPO	0.045
TTC PPO	0.039

(b) With sudden harsh braking.

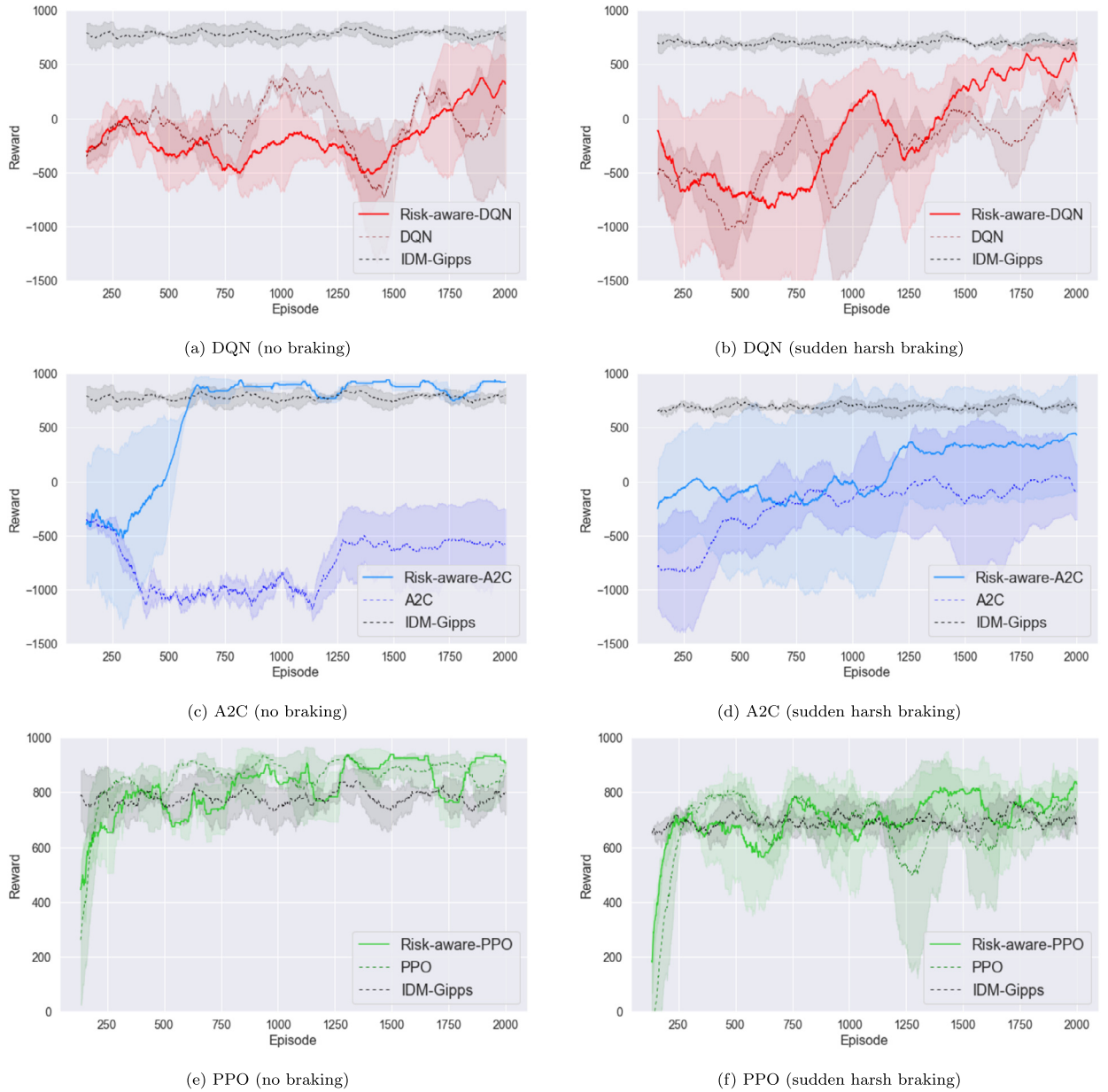


Fig. 7. Training reward plots for risk awareness experiments. All risk-aware policies (trained with RL and the collision prediction model) converged to higher reward values than their corresponding baselines (RL without the collision prediction model). The risk-aware policies converged to higher values than the rule-based IDM-Gipps model in the following cases: A2C with no sudden braking, and PPO both with and without sudden braking.

Table 10

Run-time of policies (average step time (s)).

RL policies	DQN	A2C	PPO
No collision model	0.061	0.034	0.042
Gaussian	0.098	0.078	0.083
Althoff	0.0895	0.063	0.074
TTC	0.070	0.038	0.049
Rule-based IDM-Gipps		0.0249	

well beyond the required step time of 0.2 s (sampling rate of 5 Hz), and the performance is significantly improved, as shown previously.

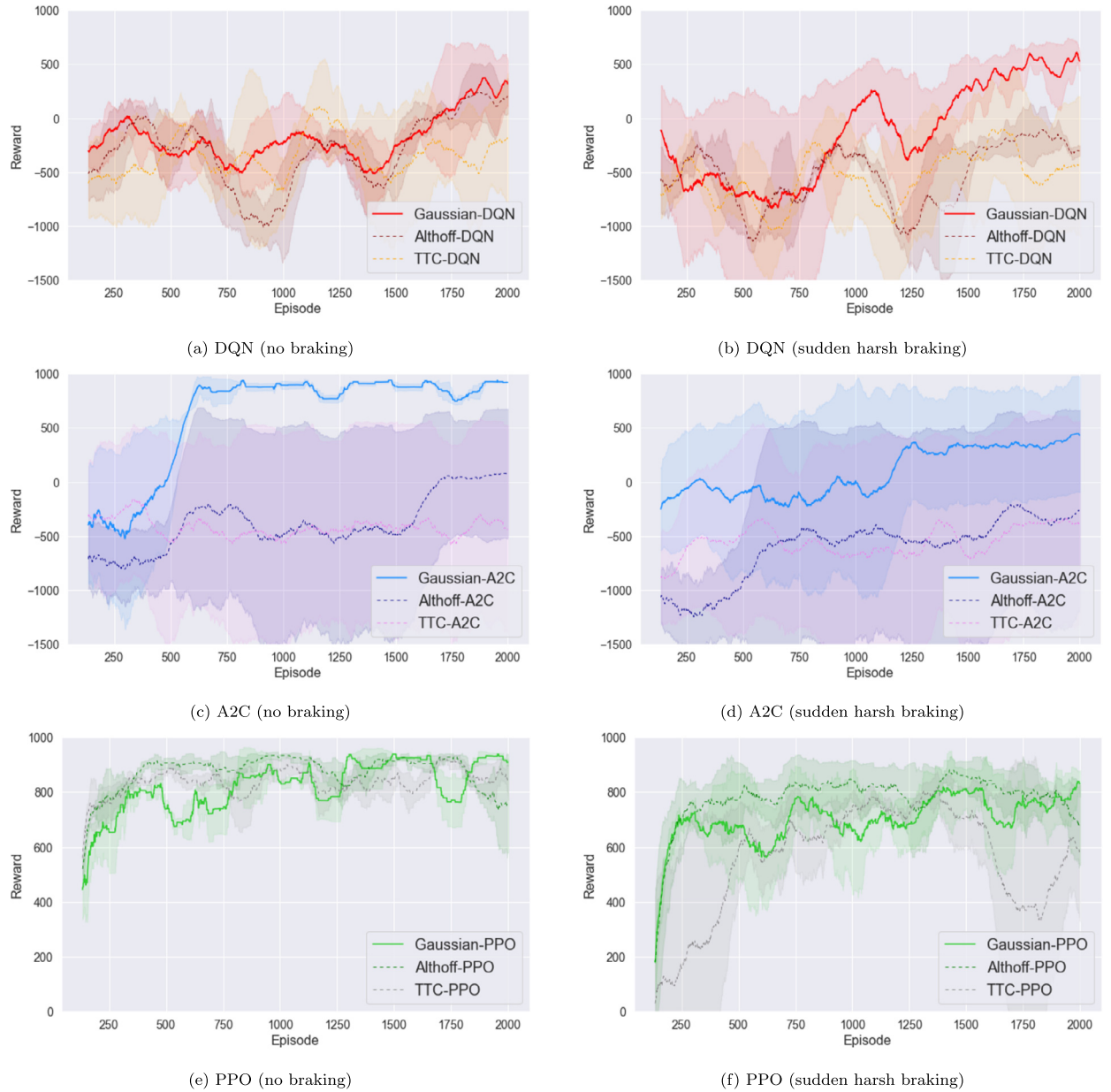


Fig. 8. Training reward plots for experiments with RL and different collision prediction models. When using DQN and A2C, the policies trained with the proposed Gaussian collision prediction model converge to a higher reward value, followed by the ones trained with Althoff's model. Policies trained with PPO behave similarly during training for the three collision prediction models.

6. Conclusions

This paper shows that the use of model-based risk functions together with AI has enormous potential for achieving higher safety and performance levels in safety-sensitive applications, and also provides post-hoc explainability in control systems.

We present a novel framework for risk-aware decision-making for AVs, which employs a bespoke dynamics-based collision prediction model together with RL. The proposed framework was implemented on a simulator for training policies using various state-of-the-art RL algorithms. The generality of the framework allow AVs to simultaneously and autonomously control various driving functions, namely adaptive cruise control, merging, lane changing, collision avoidance and emergency braking.

The evaluation experiments show that adding the proposed collision prediction model to the state vector led to a significant reduction in the collision rates. Moreover, our collision prediction model proved to be safer than other collision

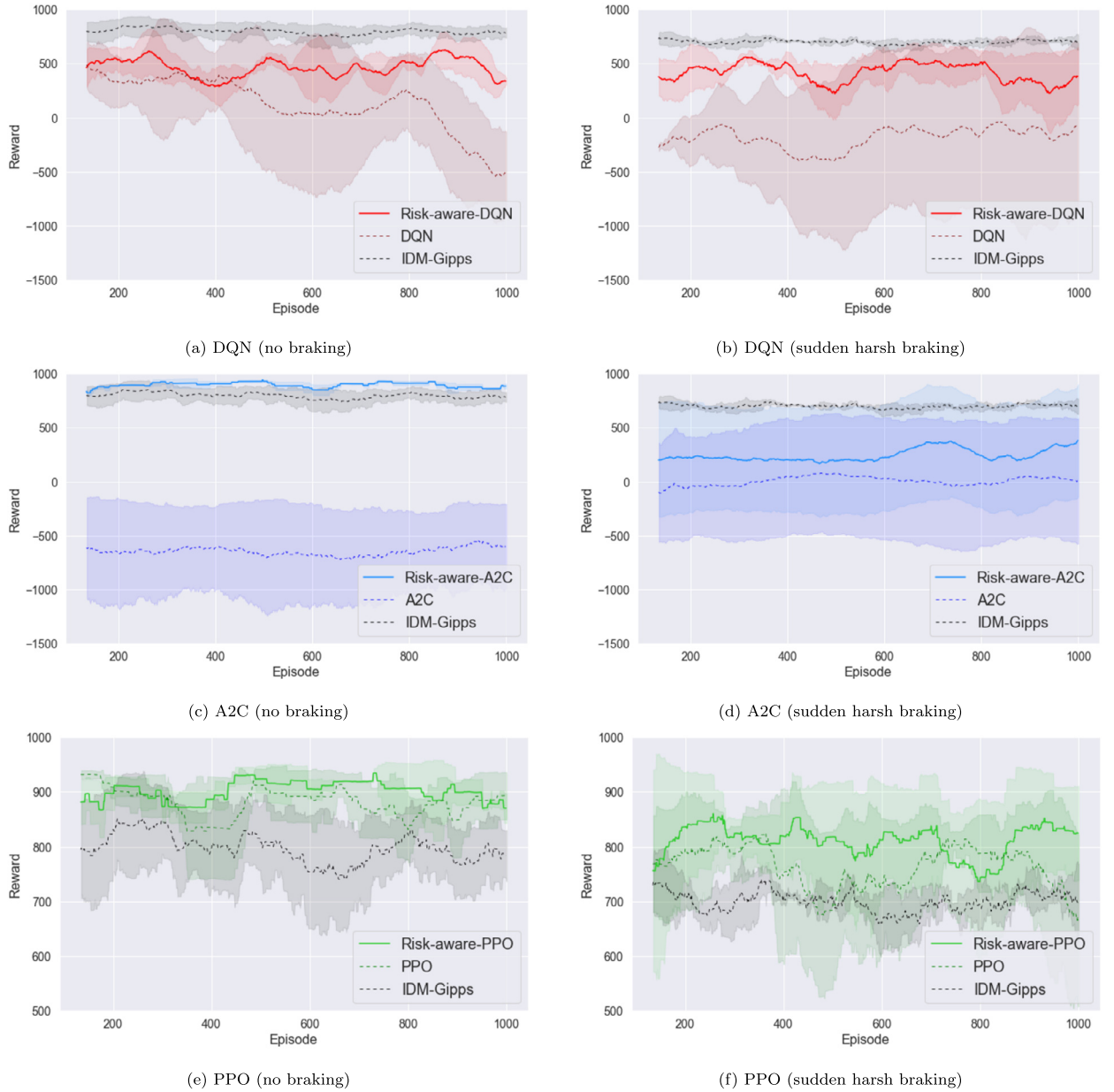


Fig. 9. Evaluation reward plots for risk awareness experiments. All risk-aware policies (trained with RL and the collision prediction model) achieved higher rewards than their corresponding baselines (RL without the collision prediction model). The risk-aware policies performed better than the rule-based IDM-Gipps model in the following cases: A2C with no sudden braking, and PPO both with and without sudden braking.

models. Our AI-based framework also outperformed a commonly used rule-based method by achieving lower collision rates, higher speeds, and shortest following distances. This translates to safer vehicle journeys, higher speeds and reduced traffic congestion.

It is important to mention some of the limitations of the presented method. If the collision prediction parameters are not properly selected, the method could fail in some scenarios. Also, as shown by the experimental results, some RL algorithms can fail to converge to an optimal policy. The problem of partial or inaccurate sensor information that could affect the state vector is not considered. Lastly, the proposed reward function was manually selected and we leave the optimization of the reward function design as future work.

Other future research lines include extending the introduced framework for solving other driving functions such as steering control, parking, or platooning. This could be done by adjusting the state space, action spaces and reward function accordingly. Additionally, the state representation and reward function in the training environment can be expanded to account for other performance metrics such as energy consumption, ride comfort, and other safety and risk functions. On

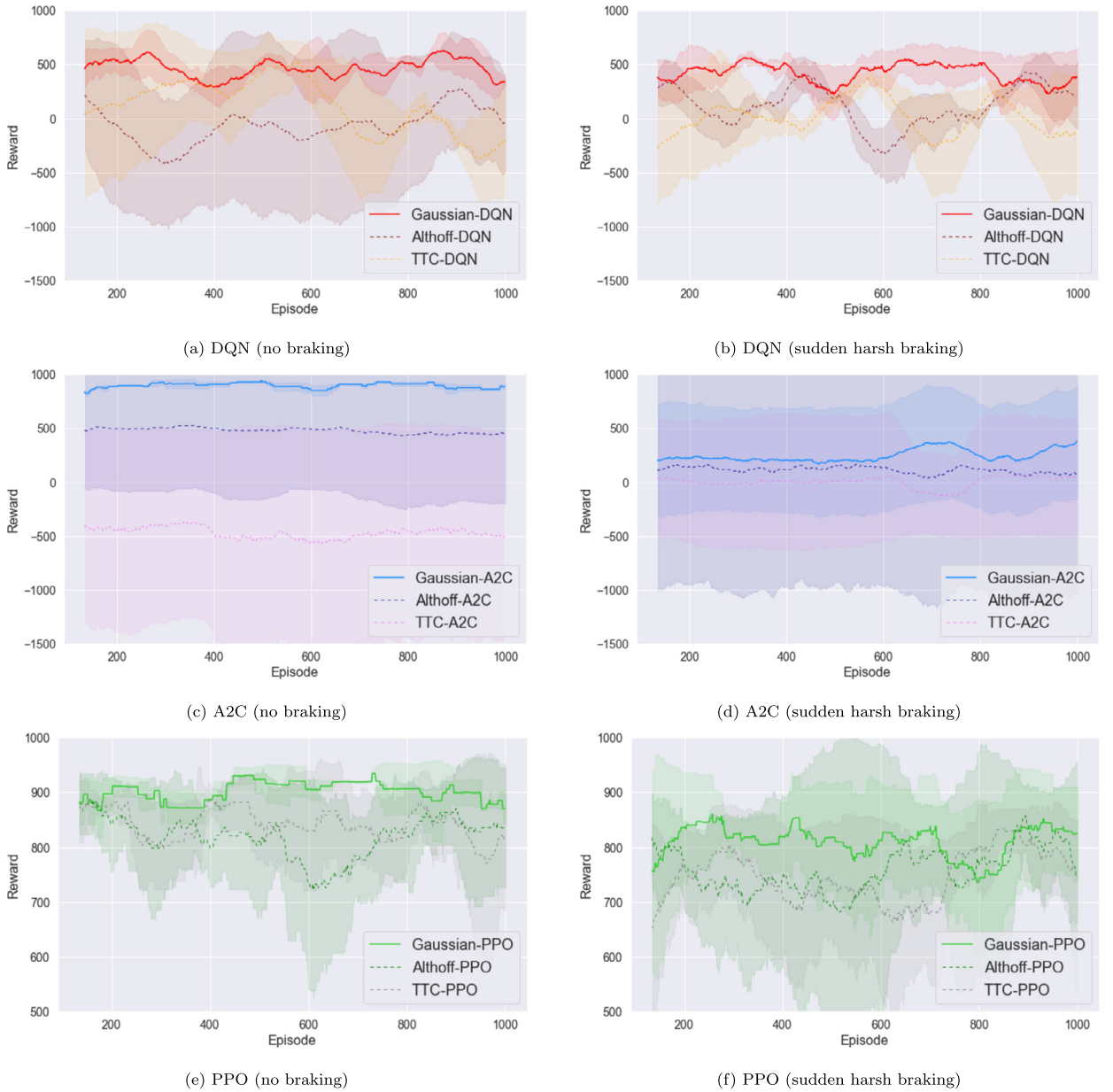


Fig. 10. Evaluation reward plots for experiments with RL and different collision prediction models. All policies trained with the proposed Gaussian collision prediction model achieve higher rewards than with Althoff's model and TTC. Althoff's model performs better than TTC when using A2C.

the other hand, incorporating multi-agent reinforcement learning and communication with infrastructure (V2X) could be advantageous.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

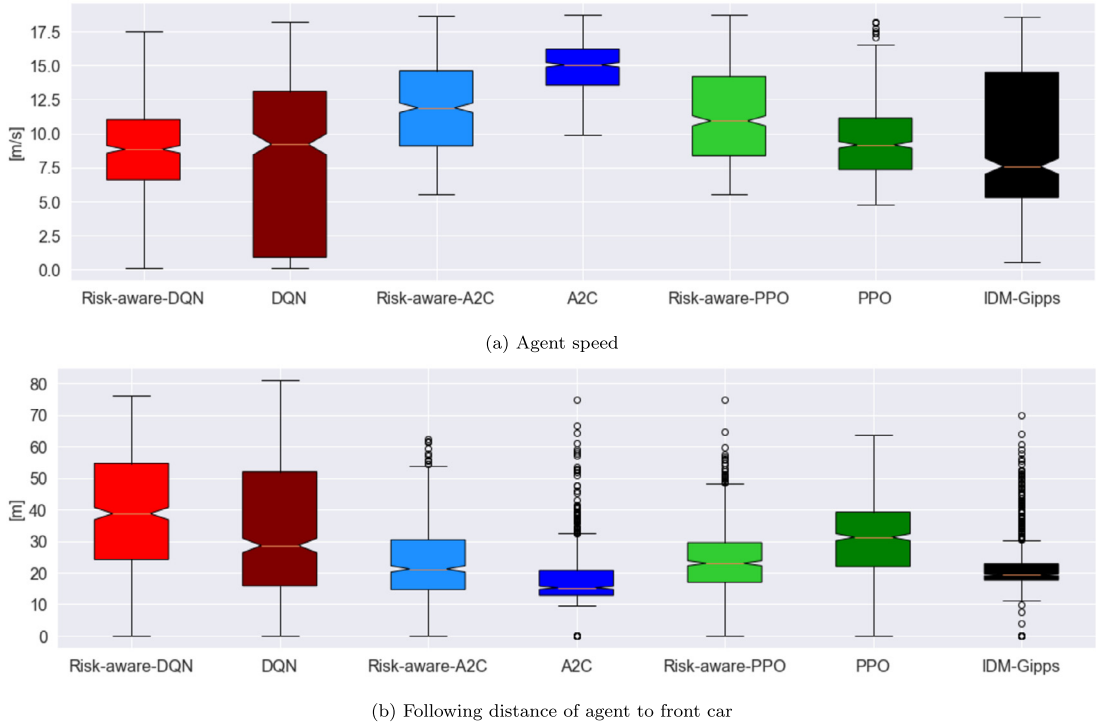


Fig. 11. Risk-aware experiments without sudden harsh braking: distribution of speed and following distance. Risk-aware policies trained with DQN and A2C present more conservative driving than their counterparts without collision prediction. However, risk-aware PPO was able to attain better rewards with higher speeds and lower following distances, while having better collision rates. IDM-Gipps has similar following distances than risk-aware PPO, but slower speeds.

Acknowledgement

This research was supported by the President's Scholarship Programme funded by Imperial College London; the Chilean National Agency for Research and Development (ANID) through the "BECAS DOCTORADO EN EXTRANJERO" program, Grant No. 72210279; Innovate UK; and a RAEng Chair in Emerging Technologies.

Appendix A. Vehicle driver models

The Gipps model is a well-known car following approach that corresponds to the safe-distance class of models. We chose this model because it considers the risk of potential collision between vehicles and is simple to calibrate [79]. In this model, the actions to change lanes are governed by a fixed set of rules that considers the risk of collision, the presence of vehicles in the other lane, and the potential gain in the average speed of the decision. The main disadvantage of this model is that it does not consider vehicle perception [80]. In addition, minor variations on the leader's vehicle can heavily affect the follower's reaction.

Gipps' model determines the speed of each vehicle at time t by applying two constraints on the acceleration: the desired speed must not be surpassed and the critical safe distance should be maintained. The acceleration and deceleration can be expressed by the following equations:

$$v_n^{acc}(t+T) = v_n(t) + 2.5 \cdot a_n \cdot T \left(1 - \frac{v_n(t)}{v_n^d}\right) \cdot \sqrt{0.025 + \frac{v_n(t)}{v_n^d}} \quad (\text{A.1})$$

$$v_n^{decc}(t+T) = -T \cdot d_n + \sqrt{T^2 \cdot d_n^2 + d_n \{2x_{n-1}(t) - x_n(t) - (S_{n-1}) - T \cdot v_n(t) + \frac{v_{n-1}(t)^2}{d'_{n-1}}\}} \quad (\text{A.2})$$

where n and $n-1$ are the follower and leader, respectively, T is the reaction time, $v_n(t)$ and $v_{n-1}(t)$ are the speeds of the follower and leader respectively at time t . v_n^d and a_n are the follower's maximum desired speed and maximum acceleration, respectively. d_n and d'_{n-1} are the most severe braking capability and $x_n(t)$ and $x_{n-1}(t)$ are the longitudinal position of the follower and leader at time t . S_{n-1} is the distance between vehicles at a stop. For modeling lane-changing decisions, we use the Gipps lane-changing model [81]. This model is a commonly used rule-based algorithm for describing realistic driving behavior. It provides asymmetric acceleration/deceleration and assumes that the vehicle speed is estimated based on the

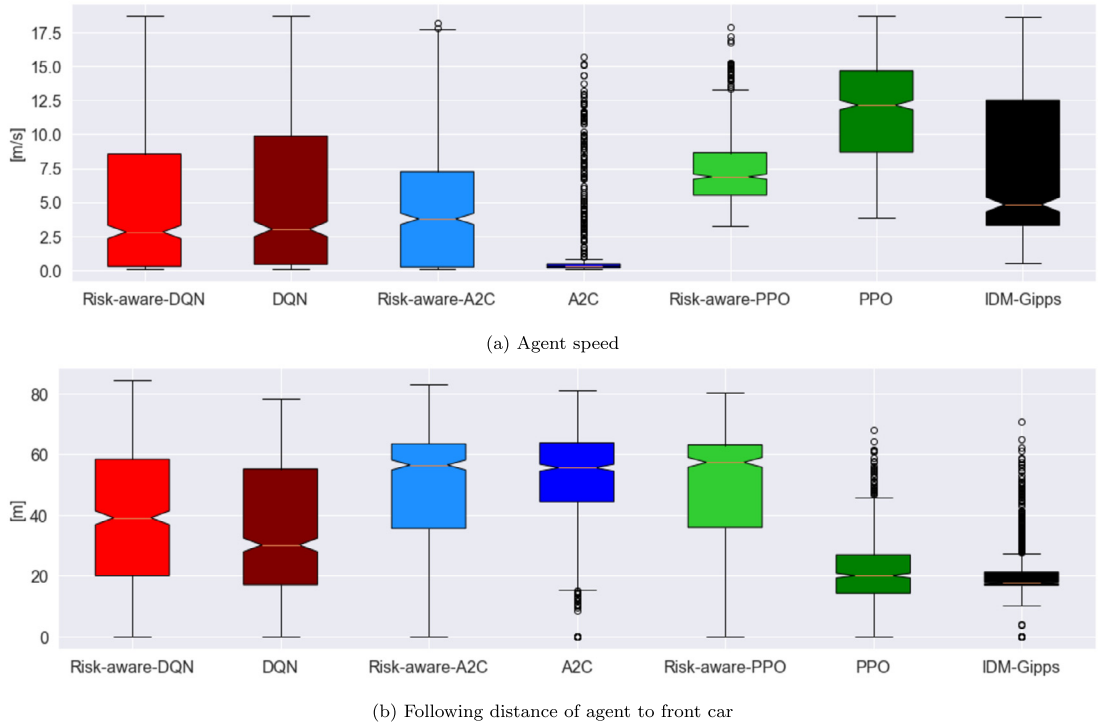


Fig. 12. Risk-aware experiments with sudden harsh braking: distribution of speed and following distance. Risk-aware policies trained with DQN and PPO present more conservative driving than their counterparts without collision prediction. A2C has converged to the local optimal policy of stopping to avoid collisions (which is sub-optimal globally). The rule-based IDM-Gipps model present a high variance in speed, and low flexibility in following distance, while risk-aware PPO adapts better with higher speeds, less variance in speed, and more robust and flexible following distances.

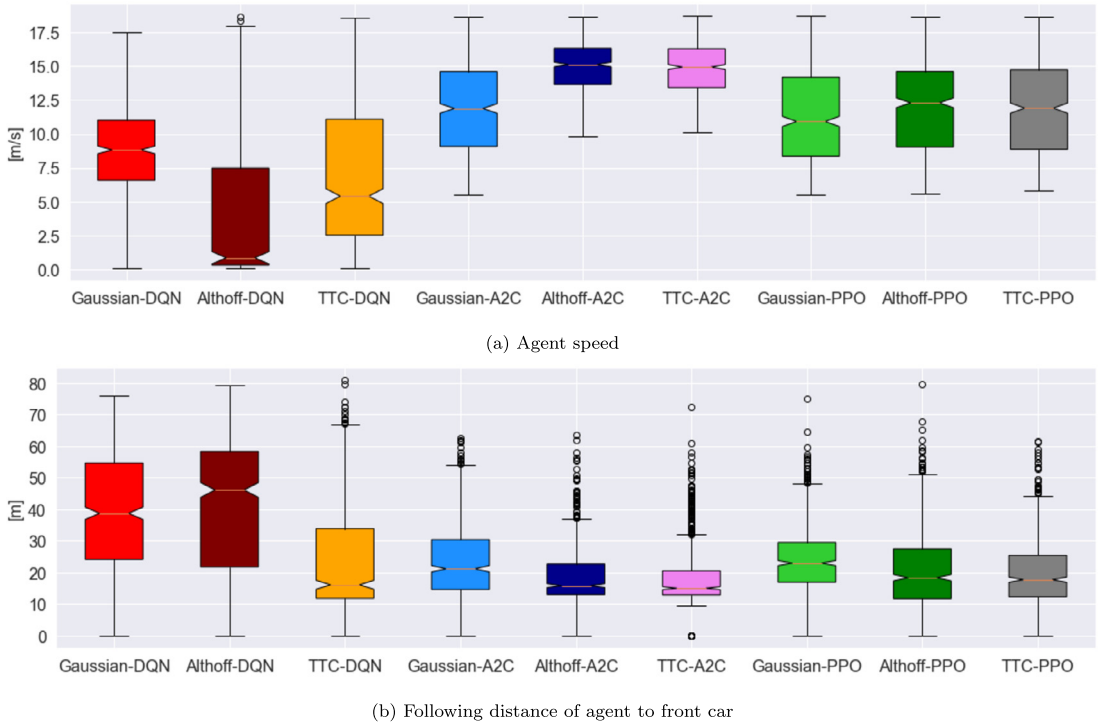


Fig. 13. Comparison of collision prediction models without sudden harsh braking: distribution of speed and following distance. Policies trained with DQN seem to present the most conservative driving behavior, while the A2C policies are the most aggressive. PPO policies achieve a better balance between speed and following distance, with Gaussian-PPO obtaining the best rewards and safety by being slightly more conservative than the other PPO policies.

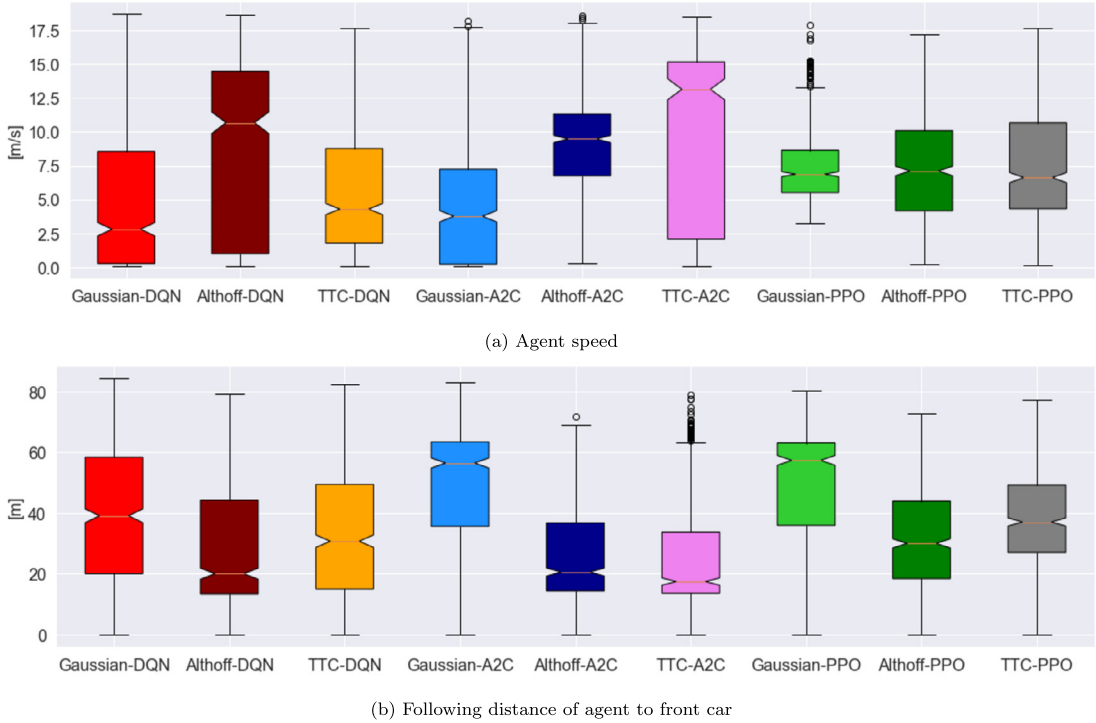


Fig. 14. Comparison of collision prediction models with sudden harsh braking: distribution of speed and following distance. Policies with the Gaussian model are more conservative (lower speeds and higher following distances) for all RL models. This shows that the Gaussian model enables the best adaptation to sudden harsh braking events.

car in front to be able to come to a complete stop if needed. The implemented version of Gipps' model uses safe following distances set according to the RSS model introduced in [82].

Appendix B. Path planning

We implemented a path planning algorithm for both the ego vehicle and the other simulated vehicles. It relies on waypoints that are set equidistantly along the centerlines of the lanes of a road. Each waypoint is connected to the set of next possible waypoints, allowing for lane changing and merging. A path for a vehicle is generated by getting the shortest path between a given origin waypoint and destination waypoint. Any shortest path algorithm or heuristic can be utilized; Dijkstra's algorithm is a common choice for shortest pathfinding. Lane changes are determined with the Gipps lane-changing model described in the previous subsection.

The paths of the other simulated agents are obtained by constructing set I^o with all possible paths $i \in I^o$ for agent o at time-step t , using the waypoint layout of the road. The probabilities q_i of paths $i \in I^o$ are calculated with the Autonomous Multiple Model Estimation (AMM) algorithm [46], with the formula

$$q_i^{(t)} = \frac{q_i^{(t-1)} L_i^{(t)}}{\sum_{j \in I^o} q_j^{(t-1)} L_j^{(t)}} \quad (\text{B.1})$$

where $L_i^{(t)}$ is the likelihood of agent taking path i at time-step t , which is initialized uniformly among all possible paths.

Appendix C. Experiments for choosing prediction times in state vector

The future collision prediction time values in the state vector were selected according to the criteria listed in Section 4.3.1. The combination of values that resulted in the lowest collision rates was {0.6 s, 1.0 s, 1.4 s}, as can be seen in Table C.11.

As in all the other experiments, the policies were trained during 2000 episodes, and evaluated over 1000 episodes; 5 random seeds were used for statistical purposes. Options with 4 or more future time values were not considered, since the average step computation time was larger than the 0.1 s requirement. Only state vectors with 3 prediction times are shown, since they achieved better results than when using 1 or 2 prediction times.

Table C.11

Collision rates for choices of collision prediction times in state vector.

Future times (s)	DQN	A2C	PPO
{0.4, 0.8, 1.2}	0.113	0.037	0.014
{0.4, 0.8, 1.4}	0.109	0.037	0.014
{0.4, 0.8, 1.6}	0.11	0.038	0.013
{0.4, 0.8, 1.8}	0.113	0.039	0.014
{0.4, 0.8, 2.0}	0.117	0.034	0.013
{0.4, 1.0, 1.4}	0.114	0.031	0.015
{0.4, 1.0, 1.6}	0.109	0.029	0.014
{0.4, 1.0, 1.8}	0.107	0.028	0.012
{0.4, 1.0, 2.0}	0.105	0.026	0.011
{0.4, 1.2, 1.6}	0.108	0.025	0.010
{0.4, 1.2, 1.8}	0.104	0.025	0.010
{0.4, 1.2, 2.0}	0.106	0.023	0.010
{0.4, 1.4, 1.6}	0.104	0.021	0.010
{0.4, 1.4, 1.8}	0.103	0.022	0.009
{0.4, 1.4, 2.0}	0.106	0.02	0.009
{0.4, 1.6, 2.0}	0.106	0.018	0.009
{0.6, 1.0, 1.4}	0.105	0.015	0.008
{0.6, 1.0, 1.6}	0.106	0.016	0.008
{0.6, 1.0, 1.8}	0.107	0.017	0.009
{0.6, 1.0, 2.0}	0.106	0.016	0.009
{0.6, 1.2, 1.6}	0.105	0.018	0.009
{0.6, 1.2, 1.8}	0.109	0.02	0.009
{0.6, 1.2, 2.0}	0.111	0.022	0.010
{0.6, 1.4, 1.6}	0.107	0.022	0.009
{0.6, 1.4, 1.8}	0.108	0.023	0.009
{0.6, 1.4, 2.0}	0.11	0.025	0.010
{0.6, 1.6, 2.0}	0.113	0.028	0.010
{0.8, 1.2, 1.6}	0.111	0.029	0.010
{0.8, 1.2, 1.8}	0.112	0.03	0.010
{0.8, 1.2, 2.0}	0.116	0.033	0.010
{0.8, 1.4, 1.6}	0.114	0.032	0.010
{0.8, 1.4, 1.8}	0.118	0.033	0.011
{0.8, 1.4, 2.0}	0.115	0.035	0.011
{0.8, 1.6, 2.0}	0.113	0.039	0.011
{1.0, 1.4, 1.6}	0.118	0.038	0.011
{1.0, 1.4, 1.8}	0.119	0.043	0.012
{1.0, 1.4, 2.0}	0.124	0.047	0.012
{1.0, 1.6, 2.0}	0.122	0.046	0.013
{1.2, 1.6, 2.0}	0.127	0.044	0.014
{1.4, 1.6, 2.0}	0.124	0.044	0.014

References

- [1] World Health Organization (WHO), Road traffic injuries, <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>, 2020. (Accessed 27 March 2021), online.
- [2] S. Chen, M. Kuhn, K. Prettnner, D.E. Bloom, The global macroeconomic burden of road injuries: estimates and projections for 166 countries, *Lancet Planet. Health* 3 (9) (2019) e390–e398.
- [3] S. Taxonomy, Definitions for Terms Related to on-Road Motor Vehicle Automated Driving Systems, Society of Automotive Engineers (SAE), Troy, MI, USA, 2014.
- [4] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, *Nature* 529 (7587) (2016) 484–489.
- [5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, *Nature* 550 (7676) (2017) 354–359.
- [6] P.R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T.J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, et al., Outracing champion gran turismo drivers with deep reinforcement learning, *Nature* 602 (7896) (2022) 223–228.
- [7] P. Wang, C.-Y. Chan, Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2017, pp. 1–6.
- [8] T. Ren, Y. Xie, L. Jiang, Cooperative highway work zone merge control based on reinforcement learning in a connected and automated environment, *Transp. Res. Rec.* 2674 (10) (2020) 363–374.
- [9] S. Triest, A. Villafior, J.M. Dolan, Learning highway ramp merging via reinforcement learning with temporally-extended actions, in: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE, pp. 1595–1600.
- [10] L. Le Mero, D. Yi, M. Dianati, A. Mouzakitis, A survey on imitation learning techniques for end-to-end autonomous vehicles, *IEEE Trans. Intell. Transp. Syst.* (2022).
- [11] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E.A. Theodorou, B. Boots, Imitation learning for agile autonomous driving, *Int. J. Robot. Res.* 39 (2–3) (2020) 286–302.
- [12] J. Garcia, F. Fernández, A comprehensive survey on safe reinforcement learning, *J. Mach. Learn. Res.* 16 (1) (2015) 1437–1480.
- [13] W. Samek, G. Montavon, A. Vedaldi, L.K. Hansen, K.-R. Müller, Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, vol. 11700, Springer Nature, 2019.

- [14] M. Cummings, Adaptation of human licensing examinations to the certification of autonomous systems, in: *Safe, Autonomous and Intelligent Vehicles*, Springer, 2019, pp. 145–162.
- [15] A.B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bénéttot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Explainable artificial intelligence (xai): concepts, taxonomies, opportunities and challenges toward responsible ai, *Inf. Fusion* 58 (2020) 82–115.
- [16] S. Ulbrich, A. Reschka, J. Rieken, S. Ernst, G. Bagschik, F. Dierkes, M. Nolte, M. Maurer, Towards a functional system architecture for automated vehicles, *arXiv preprint*, arXiv:1703.08557, 2017.
- [17] W. Schwarting, J. Alonso-Mora, D. Rus, Planning and decision-making for autonomous vehicles, *Annu. Rev. Control, Robot. Autonom. Syst.* 1 (2018) 187–210.
- [18] R. Chopra, S.S. Roy, End-to-end reinforcement learning for self-driving car, in: *Advanced Computing and Intelligent Engineering*, Springer, Singapore, 2020, pp. 53–61.
- [19] D.A. Pomerleau, *Alvin: an autonomous land vehicle in a neural network*, Tech. Rep., Carnegie-Mellon Univ Pittsburgh PA Artificial Intelligence and Psychology..., 1989.
- [20] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, *arXiv preprint*, arXiv:1604.07316, 2016.
- [21] D. Wakabayashi, Self-driving uber car kills pedestrian in Arizona, where robots roam, *N.Y. Times* 19 (03) (2018).
- [22] N. Boudette, N. Choksi, Us will investigate tesla's autopilot system over crashes with emergency vehicles, *N.Y. Times* (2021).
- [23] M. Heger, Consideration of risk in reinforcement learning, in: *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 105–111.
- [24] A. Nilim, L. El Ghauui, Robust control of Markov decision processes with uncertain transition matrices, *Oper. Res.* 53 (5) (2005) 780–798.
- [25] H.M. Markowitz, G.P. Todd, *Mean-Variance Analysis in Portfolio Choice and Capital Markets*, vol. 66, John Wiley & Sons, 2000.
- [26] M. Sato, H. Kimura, S. Kobayashi, Td algorithm for the variance of return and mean-variance reinforcement learning, *Trans. Jpn. Soc. Artif. Intell.* 16 (3) (2001) 353–362.
- [27] P. Geibel, F. Wyszotzki, Risk-sensitive reinforcement learning applied to control under constraints, *J. Artif. Intell. Res.* 24 (2005) 81–108.
- [28] T.M. Moldovan, P. Abbeel, Safe exploration in Markov decision processes, *arXiv preprint*, arXiv:1205.4810, 2012.
- [29] K. Driessens, S. Džeroski, Integrating guidance into relational reinforcement learning, *Mach. Learn.* 57 (3) (2004) 271–304.
- [30] P. Abbeel, A. Coates, A.Y. Ng, Autonomous helicopter aerobatics through apprenticeship learning, *Int. J. Robot. Res.* 29 (13) (2010) 1608–1639.
- [31] A.L. Thomaz, C. Breazeal, Teachable robots: understanding human teaching behavior to build more effective robot learners, *Artif. Intell.* 172 (6–7) (2008) 716–737.
- [32] C. Gehring, D. Precup, Smart exploration in reinforcement learning using absolute temporal difference errors, in: *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, 2013, pp. 1037–1044.
- [33] J. Garcia, F. Fernández, Safe exploration of state and action spaces in reinforcement learning, *J. Artif. Intell. Res.* 45 (2012) 515–564.
- [34] J.C. Hayward, Near miss determination through use of a scale of danger, 1972.
- [35] S.G. Klauer, T.A. Dingus, V.L. Neale, J.D. Sudweeks, D.J. Ramsey, et al., The impact of driver inattention on near-crash/crash risk: an analysis using the 100-car naturalistic driving study data, 2006.
- [36] J. Wang, Y. Zheng, X. Li, C. Yu, K. Kodaka, K. Li, Driving risk assessment using near-crash database through data mining of tree-based model, *Accid. Anal. Prev.* 84 (2015) 54–64.
- [37] D. Åsljung, J. Nilsson, J. Fredriksson, Using extreme value theory for vehicle level safety validation and implications for autonomous vehicles, *IEEE Trans. Intell. Veh.* 2 (4) (2017) 288–297.
- [38] M. Strickland, G. Fainekos, H.B. Amor, Deep predictive models for collision risk assessment in autonomous driving, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 4685–4692.
- [39] C. Chen, H. Xiang, T. Qiu, C. Wang, Y. Zhou, V. Chang, A rear-end collision prediction scheme based on deep learning in the Internet of vehicles, *J. Parallel Distrib. Comput.* 117 (2018) 192–204.
- [40] D. Lee, H. Yeo, Real-time rear-end collision-warning system using a multilayer perceptron neural network, *IEEE Trans. Intell. Transp. Syst.* 17 (11) (2016) 3087–3097.
- [41] L. Fraade-Blumar, M.S. Blumenthal, J.M. Anderson, N. Kalra, *Measuring automated vehicle safety: Forging a framework*, 2018.
- [42] M. Althoff, O. Stursberg, M. Buss, Model-based probabilistic collision detection in autonomous driving, *IEEE Trans. Intell. Transp. Syst.* 10 (2) (2009) 299–310.
- [43] A. Lambert, D. Gruyer, G. Saint Pierre, A.N. Ndjeng, Collision probability assessment for speed control, in: *2008 11th International IEEE Conference on Intelligent Transportation Systems*, IEEE, 2008, pp. 1043–1048.
- [44] J. Strohbeck, V. Belagiannis, J. Müller, M. Schreiber, M. Herrmann, D. Wolf, M. Buchholz, Multiple trajectory prediction with deep temporal and spatial convolutional neural networks, in: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 1992–1998.
- [45] H. He, H. Dai, N. Wang, Ust: unifying spatio-temporal context for trajectory prediction in autonomous driving, in: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 5962–5969.
- [46] X.R. Li, V.P. Jilkov, Survey of maneuvering target tracking. Part V. Multiple-model methods, *IEEE Trans. Aerosp. Electron. Syst.* 41 (4) (2005) 1255–1321.
- [47] E. Candela, Y. Feng, D. Mead, Y. Demiris, P. Angeloudis, Fast collision prediction for autonomous vehicles using a stochastic dynamics model, in: *IEEE ITSC 2021*, 2021.
- [48] L. Pannocchi, C. Di Franco, M. Marinoni, G. Buttazzo, Integrated framework for fast prototyping and testing of autonomous systems, *J. Intell. Robot. Syst.* 96 (2) (2019) 223–243, <https://doi.org/10.1007/s10846-018-0969-3>, <http://link.springer.com/10.1007/s10846-018-0969-3>.
- [49] J. Rios-Torres, A.A. Malikopoulos, Automated and cooperative vehicle merging at highway on-ramps, *IEEE Trans. Intell. Transp. Syst.* 18 (4) (2016) 780–789.
- [50] B. Van Arem, C.J. Van Driel, R. Visser, The impact of cooperative adaptive cruise control on traffic-flow characteristics, *IEEE Trans. Intell. Transp. Syst.* 7 (4) (2006) 429–436.
- [51] X.-m. Chen, M. Jin, C.-Y. Chan, Y.-s. Miao, J.-w. Gong, Bionic decision-making analysis during urban expressway ramp merging for autonomous vehicle, *Tech. Rep.*, 2017.
- [52] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al., Autonomous driving in urban environments: boss and the urban challenge, *J. Field Robot.* 25 (8) (2008) 425–466.
- [53] T. Tram, A. Jansson, R. Grönberg, M. Ali, J. Sjöberg, Learning negotiating behavior between cars in intersections using deep q-learning, in: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3169–3174.
- [54] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, K. Fujimura, Navigating occluded intersections with autonomous vehicles using deep reinforcement learning, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2034–2039.
- [55] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, Y. Zhao, Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles, *IEEE Trans. Neural Netw. Learn. Syst.* 32 (12) (2021) 5435–5444.
- [56] L. Wen, J. Duan, S.E. Li, S. Xu, H. Peng, Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization, in: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 1–7.
- [57] C. Skoglund, Risk-aware autonomous driving using pomdps and responsibility sensitive safety, Master's thesis, KTH Royal Institute of Technology, 2021.

- [58] T. Nyberg, C. Pek, L. Dal Col, C. Norén, J. Tumova, Risk-aware motion planning for autonomous vehicles with safety specifications, in: 2021 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2021, pp. 1016–1023.
- [59] M. Fallgren, M. Dillinger, T. Mahmoodi, T. Svensson, Cellular V2X for Connected Automated Driving, Wiley Online Library, 2021.
- [60] A. Heuillet, F. Couthouis, N. Díaz-Rodríguez, Explainability in deep reinforcement learning, *Knowl.-Based Syst.* 214 (2021) 106685.
- [61] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
- [62] X. Koutsoukos, D. Riley, Computational methods for reachability analysis of stochastic hybrid systems, in: International Workshop on Hybrid Systems: Computation and Control, Springer, 2006, pp. 377–391.
- [63] C. Hubmann, M. Becker, D. Althoff, D. Lenz, C. Stiller, Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles, in: 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2017, pp. 1671–1678.
- [64] S.V. Albrecht, C. Brewitt, J. Wilhelm, B. Geyvner, F. Eiras, M. Dobre, S. Ramamoorthy, Interpretable goal-based prediction and planning for autonomous driving, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 1043–1049.
- [65] H. Chae, C.M. Kang, B. Kim, J. Kim, C.C. Chung, J.W. Choi, Autonomous braking system via deep reinforcement learning, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2017, pp. 1–6.
- [66] C. Desjardins, B. Chaib-Draa, Cooperative adaptive cruise control: a reinforcement learning approach, *IEEE Trans. Intell. Transp. Syst.* 12 (4) (2011) 1248–1260.
- [67] P. Bokare, A. Maurya, Acceleration-deceleration behaviour of various vehicle types, *Transp. Res. Proc.* 25 (2017) 4733–4749.
- [68] Y. Tang, S. Agrawal, Discretizing continuous action space for on-policy optimization, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 5981–5988.
- [69] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint, arXiv:1312.5602, 2013.
- [70] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1928–1937.
- [71] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint, arXiv:1707.06347, 2017.
- [72] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
- [73] F. Rosique, P.J. Navarro, C. Fernández, A. Padilla, A systematic review of perception system and simulators for autonomous vehicles research, *Sensors* 19 (3) (2019) 648.
- [74] N. Raju, S. Arkatkar, G. Joshi, Evaluating performance of selected vehicle following models using trajectory data under mixed traffic conditions, *J. Intell. Transp. Syst.* 24 (6) (2020) 617–634.
- [75] M. Treiber, A. Hennecke, D. Helbing, Congested traffic states in empirical observations and microscopic simulations, *Phys. Rev. E* 62 (2) (2000) 1805.
- [76] P.G. Gipps, A behavioural car-following model for computer simulation, *Transp. Res., Part B, Methodol.* 15 (2) (1981) 105–111.
- [77] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, L. Eckstein, The ind dataset: a drone dataset of naturalistic road user trajectories at German intersections, 2019.
- [78] NHTSA, Traffic safety facts annual report tables, <https://cdan.nhtsa.gov/tsftables/tsfar.html>.
- [79] H.U. Ahmed, Y. Huang, P. Lu, A review of car-following models and modeling tools for human and autonomous-ready driving behaviors in micro-simulation, *Smart Cities* 4 (1) (2021) 314–335, <https://doi.org/10.3390/smartcities4010019>, <https://www.mdpi.com/2624-6511/4/1/19>.
- [80] H. Lazar, K. Rhoulami, M.D. Rahmani, A review analysis of optimal velocity models, *Period. Polytech., Transp. Eng.* 44 (2016) 123–131, <https://doi.org/10.3311/PPtr.8753>.
- [81] P.G. Gipps, A model for the structure of lane-changing decisions, *Transp. Res., Part B, Methodol.* 20 (5) (1986) 403–414, [https://doi.org/10.1016/0191-2615\(86\)90012-3](https://doi.org/10.1016/0191-2615(86)90012-3), <https://www.sciencedirect.com/science/article/pii/0191261586900123>.
- [82] S. Shalev-Shwartz, S. Shammah, A. Shashua, On a formal model of safe and scalable self-driving cars, <https://rssresearch.mobileye.com/t/resources-on-rss/33>.