# Trajectory Planning for Autonomous Vehicle Using Iterative Reward Prediction in Reinforcement Learning

Hyunwoo Park[1]

***Abstract*— Traditional trajectory planning methods for autonomous vehicles have several limitations. For example, heuristic and explicit simple rules limit generalizability and hinder complex motions. These limitations can be addressed using reinforcement learning-based trajectory planning. However, reinforcement learning suffers from unstable learning and existing reinforcement learning-based trajectory planning methods do not consider the uncertainties. Thus, this paper, proposes a reinforcement learning-based trajectory planning method for autonomous vehicles. The proposed method includes an iterative reward prediction method that stabilizes the learning process, and an uncertainty propagation method that makes the reinforcement learning agent aware of uncertainties. The proposed method was evaluated using the CARLA simulator. Compared to the baseline methods, the proposed method reduced the collision rate by 60.17%, and increased the average reward by 30.82 times. A video of the proposed method is available at `https://www.youtube.com/watch?v=PfDbaeLfcN4`.**

***Index terms*—-Autonomous Vehicle, Reinforcement Learning, Motion Planning**

## I. INTRODUCTION

Since the 2007 DARPA Urban Challenge [1], autonomous vehicles(AVs) have been studied intensively. In additon, planning methods for AVs have also been researched intensively and the findings have allowed AVs to drive successfully within limited areas, using rule-based and optimization-based algorithms, explicit heuristic rules, and parameters specified for the given area. However, these traditional approaches suffer from several limitations, including a lack of generality and a lack of complex motion. For example, the heuristic rules and parameters specified for the given area may not be applied to the other areas which impacts the scalability. In addition, many possible scenarios must be considered in real-world applications. If various scenarios are generalized with few scenarios(e.g., lane following, and lane changing), an overly simple policy could be obtained. Numerous studies have employed deep learning to address these limitations [2]–[17].

The most popular approach is imitation learning(IL), which learns a driving policy directly from expert driving data [2], [3]; however, IL also has several limitations, including the cost of scalability, simple driving policies, and safety. For example to scale up the AV using the IL method, expert data must be obtained for all scenarios and targeted areas, which is costly. In terms of driving policies, IL is typically used for handling simpler driving tasks, e.g., lane following. To learn complex policies or policies in corner cases, it should have a lot of data which can be costly and time-consuming. In addition, an expert's demonstrated policies for complex scenarios or corner cases are more distributed than the simple scenarios, which may yield large errors or learning may be infeasible [4]. Relative to safety limitations, insufficient amounts of data on dangerous cases or corner cases are available; thus, the IL agent could output dangerous policies due to a lack of training data.

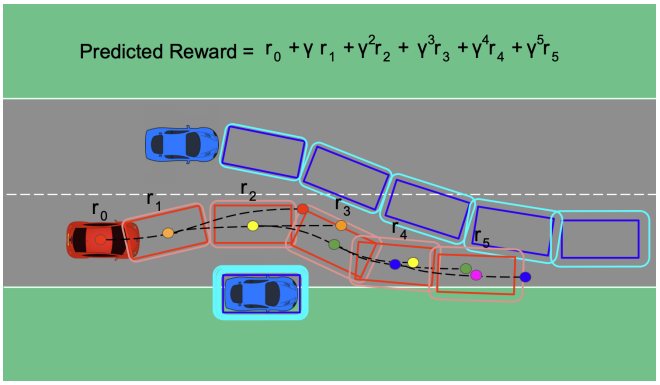


Fig. 1: The AV(red car) plans a trajectory(red boxes) along the lane while avoiding a parked car(blue car) and a car that is changing lanes(blue car) from the left lane. Other vehicles' predicted trajectories are represented as blue boxes and trajectories considering uncertainty are represented as rounded light blue boxes. The goal of each AV's trajectory is determined iteratively in each time step from the previously predicted state of the AV and the states of other vehicles. The AV's states and corresponding goals are represented as red, orange, yellow, green, blue, and purple circles(shown in chronological order). The AV's planned trajectories are shown as red boxes and trajectories considering uncertainty are shown as rounded light red boxes. The predicted states of other vehicles and the iteratively predicted states of the AV are employed for prediction of reward within the planning horizon which makes the RL learning process stable.

Another approach is the reinforcement learning(RL) method, which learns a policy via self-exploration and reinforcement without expert data. RL can also simulate and learn both complex policies and policies in corner cases. However, RL suffers from unstable learning when a neural network is used as a function approximation [18]. Error of function approximation results in unstable learning or even divergence [19], [20]. In addition, most previous RL-based trajectory planning studies did not consider the uncertainties of the object detection, trajectory prediction of other traffic participants, localization, and control modules that are essential for AV to navigate. Note that not considering uncertainties could cause sudden decelerations or even accidents.

*This work is supported by the Korea Agency for Infrastructure Technology Advancement(KAIA) grant funded by the Ministry of Land, Infrastructure and Transport. (RS-2021-KA160853, Road traffic Infrastructure monitoring and emergency recovery support service technology development)

[1] ThorDrive, Seoul, 07268, Republic of Korea

Thus this study, proposes an RL-based trajectory planning method for AVs that overcomes the identified limitations of RL and traditional planning methods. The proposed method employs a reward prediction(RP) during the learning process to stabilize the learning process. In addition, an iterative RP(IRP) method that uses RP iteratively to predicts relevant states, actions, and corresponding rewards accurately is employed. As a result, the performance of the agent and the learning stability are increased. In addition, to consider uncertainties and to drive safe, uncertainty propagation strategy [21], [22] is applied during the RP. An overview of the proposed method is shown in Fig.1.

The primary contributions of this study are summarized as follows:

- The proposed method increases learning stability and the performance of the RL agent.
- The proposed method allows the RL agent to be aware of uncertainty.
- A demonstration and comparison of the proposed method with the baseline methods in the CARLA simulator are presented.

The remainder of the paper is organized as follows: Section II reviews related works of RL-based trajectory planning methods and uncertainty-aware planning methods. Section III defines problem formulation and RP, IRP, and application of uncertainty propagation is proposed. Section IV shows how the proposed method and baseline methods are evaluated in the CARLA simulator. Section V analyzes the evaluation results and shows how the key metrics are improved. Section VI concludes the proposed method and discusses future works.

## II. Related Works

### A. RL-based Trajectory Planning

Previous RL-based trajectory planning methods for AVs can be divided into two categories according to the action of an RL agent, i.e., 1) control command and 2) the Goal of the trajectory.

*1) Control Command:* Methods that action of an RL agent is a control command [5]–[10], [14] use a lateral control(e.g., steer angle and steering rate) and a longitudinal control(e.g., acceleration and jerk) as an output of an RL agent. However, such methods tend to fail to learn easily. The variance of action affects the learning process; thus, to drive an AV successfully using a control command, very specific policies are required to yield good rewards. For example, in a highway scenario, a small turn of the steering wheel may yield catastrophic results. This specific policy requirement makes it difficult for RL agents to explore and find good states and actions, which leads to sample inefficiency during training and ineffective learning. Thus, unless the RL agent find good policy early on by chance, learning will fail. In addition, an agent's intentions are unknown; thus they lack interpretability.

Kendall et al. [5] employed monocular images as an observation and DDPG as the main algorithm to follow the lane in real-world scenarios. In addition, Chen et al. [6], employed the bird-view semantic mask as an observation and evaluated their method in CARLA simulator. Their work was developed further [7] by increasing interpretability using the probabilistic graphical model. Saxena et al. [8], employed a field-of-view as an observation and proximal policy optimization(PPO) as the main algorithm. Their primary task was lane changing in dense traffic scenarios. In addition, Wu et al. [9], employed the Dyna algorithm with PPO as the main algorithm, and they imitated the world model using the Gaussian process. Li et al. [14]. proposed a method using an hierarchical RL(HRL). Their model-based high-level policy generates subgoals via optimization that utilizes a low-level policy and an offline low-level policy outputs a control command.

*2) Goal for Trajectory:* Approaches that action of RL agent is goal/goals for trajectory [11]–[13], [15] are comparably stable in learning. The action variance of these approaches has a relatively small effect on learning process of RL compared to approaches that action of an RL agent is control command. For example, in a highway scenario, a small change in the lateral deviation of a goal would result in a smaller amount of change in a result compared to the result of the control command example.

Gao et al. [11] proposed method using HRL. Their high-level policy generates subgoals in the Frenet frame to guarantee the temporal and spatial reachability of the generated goal and the low-level policy outputs control commands. Their work was developed further [12] by ensuring safety using the safe-state enhancement method. In addition, Qiao et al. [13] employed a hybrid HRL method, where a high-level policy generates optimal behavior decisions, and a low-level policy generates a trajectory point that the AV intends to trace. They also employed a PID controller to trace the trajectory point. Ma et al. [15] used the latent state inference method, and employs PPO as a main algorithm.

### B. Uncertainty-aware Planning

Uncertainty-aware planning methods are used to plan a trajectory for an AV by considering the uncertainty of the AV(i.e., localization and control) and the traffic participants(i.e., object detection, and trajectory prediction). Xu et al. [21] employed a Kalman filter to estimate the uncertainty of the traffic participants, and the LQG framework to estimate the uncertainty of the AV. They also used the uncertainty estimation in planning by widening the size of the AV and the traffic participants when checking for the collision conditions. Fu et al. [22] and Qie et al. [23] also employed a Kalman filter to estimate the uncertainty of the traffic participant. Fu et al. used estimated uncertainty as a chance constraint when planning a velocity profile, and Qie et al. employed estimated uncertainty in a tube-based MPC to plan a trajectory. In addition, Hubmann et al. [24] formulated the planning problem with uncertainties as a partially observable Markov decision process. They estimated the intent of a traffic participant and utilized it as an uncertainty. By using the adaptive belief tree and uncertainty,

they determined the optimal longitudinal motion of the AV. Khaitan et al. [25] estimated the uncertainty of the traffic participants by utilizing reachable set in short-term horizon and used it in the tube MPC to execute the trajectory safely in the presence of uncertainty.

## III. Method

To solve trajectory planning problems using RL, RL algorithms with continuous action space, e.g., the DDPG [26], TD3 [20], and PPO [27] algorithms, are more suitable than algorithms that utilize a discrete action space. Since getting smooth behaviors requires an increase in the size of the discrete action space which leads to discrete control methods being intractable. In addition, for simplicity and interpretability, algorithms with deterministic policies(rather than stochastic policies) are selected. Furthermore, generating the goal for a trajectory is a better action choice than control command because more specific policies are required for control command methods to yield rewards successfully. Thus, the proposed trajectory planning method generates a goal using a deterministic continuous control RL algorithm, and we assume that information about the localization, route path, trajectory prediction of other traffic participants, and object detection is given.

### A. Problem Formulation

We formulate the trajectory planning problem as a Markov decision process, which is defined by the tuple $(S, A, P, R)$. Here, $s \in S$ is the continuous state space, $a \in A$ is the continuous action space, $P$ is the probability of state transition, and $R$ is the reward received after each transition. The return is defined as discounted sum of rewards $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, where $\gamma \in (0, 1)$ is the discount factor determining the priority of short-term rewards. The purpose of RL is to learn an optimal policy that maximizes the expected accumulated rewards as follows:

$$\max_{\pi} J(\pi) = \mathbb{E}_{s\sim\rho^{\pi}, a\sim\pi}[\sum_{i=0}^{\infty} \gamma^i r(s, a)], \tag{1}$$

where $r$ is the reward function, $\rho^{\pi}$ is the state distribution under the policy $\pi$.

### B. Reward Prediction

Continuous control RL algorithms employ the policy gradient method to learn policies directly. The policy gradient method maximizes the following objective function.

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{s\sim\rho^{\pi}, a\sim\pi_{\theta}}[\nabla_{\theta} log\pi_{\theta}(a|s) Q^{\pi}(s, a)]. \tag{2}$$

This theorem is derived from the following objective function:

$$J(\pi_{\theta}) = \mathbb{E}_{s\sim\rho^{\pi}, a\sim\pi_{\theta}}[r(s, a)]. \tag{3}$$

The objective function $J(\pi_{\theta})$ in (2) can be defined as the action value function $Q$, the advantage function $A$, and the TD error $\delta$. The action value and advantage functions

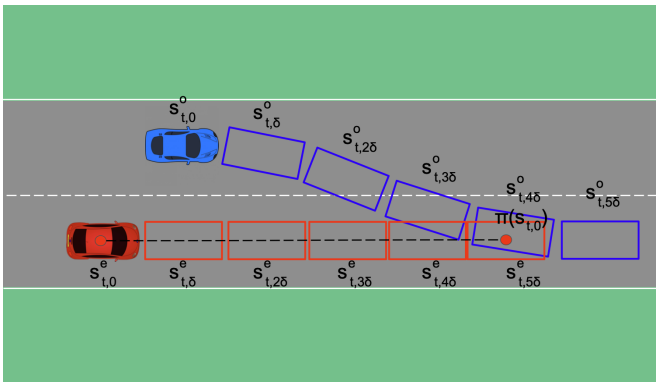


Fig. 2: The AV(red car) is following the lane while the other traffic participant(blue car) attempts to change lanes. The predicted states of the AV and the other traffic participant are represented as red and blue boxes respectively. The state of the AV at time $t$ and its goal is represented as red circles. The predicted states of the other traffic participant are assumed to be given. The predicted states(red boxes) of the AV are predicted by the RL agent's action $\pi(s_{t,0})$, i.e., the goal of the trajectory.

are approximated with neural networks in deep RL method; however, function approximation with neural network always involves errors, which causes instability during the learning process and poor performance. To improve learning stability, we utilize N-step SARSA concept. N-step SARSA improves the learning stability by utilizing the error reduction property of n-step returns. However, the rewards utilized in N-step SARSA includes randomness and variance due to the randomness of state transitions, which also causes learning instability.

We propose an RP method that utilizes the error reduction property of the n-step returns and resolves the aforementioned problem caused by variance of rewards. Rewards are predicted by predicting the expectations of future states. Unlike N-step SARSA, the expected return from the action value function of RP is not based on average rewards resulting from state transitions, but rather rewards associated with expected states. Thus, we exploit the advantages of N-step SARSA while reducing the effect by the variance of the rewards. The AV framework involves the prediction of traffic participants and the planned trajectory of the AV can be obtained using the output goal of the RL agent. Thus, future rewards can be predicted by utilizing the planned future states of the AV and the predicted states of other traffic participants. The proposed RP method utilizes the Bellman equation and is employed during the action value function update process as follows:

$$\begin{aligned}
&s^e_{t,\tau} \in \mathcal{T}^t_e,\ s^{o,k}_{t,\tau} \in \mathcal{T}^t_{o,k},\ s^o_{t,\tau} = \{s^{o,}_{t,\tau}, s^{o,1}_{t,\tau}, \cdots s^{o,n}_{t,\tau}\} \\
&s_{t,\tau} = f(s^e_{t,\tau}, s^o_{t,\tau}),\ r_{t,\tau+\delta} = g(s_{t,\tau}, s_{t,\tau+\delta}), \\
&J(\pi) = Q^{\pi}(s_t, a_t), \\
&L(\theta^Q) = \mathbb{E}_{s_t\sim\rho^{\pi}, a_t\sim\pi_{\theta}}[(Q^{\pi}(s_t, a_t) - y_t)^2], \\
&y_t = r_{t,\delta} + \gamma \cdot r_{t,2\delta} + \cdots \\
&\qquad + \gamma^{T/\delta-1} \cdot r_{t,T} + \gamma^{T/\delta} \cdot Q^{\pi}(s_{t,T}, \pi(s_{t,T})),
\end{aligned} \tag{4}$$

where $T$ is a planning/prediction horizon, $\tau$ is the time

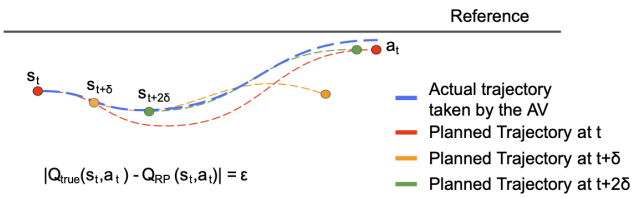


Fig. 3: Demonstration of inaccurate RP. Here, the agent has planned a trajectory that makes the AV to be close to the reference. The trajectories planned at each time step beginning from each state action pair are represented by different colors. The return predicted by RP, $Q_{RP}$, is based on the red trajectory planned at time $t$. The agent plans a new trajectory at each step; thus, the planned trajectory that RP utilized at time $t$(red) and actual trajectory executed by the AV(blue) are different. Thus, $Q_{RP}$ has an error of $\epsilon$ between the true return $Q_{true}$ which is a return obtained by following the blue trajectory.

within the planning horizon $\tau \in [0, T]$, $s^e_{t,\tau}$ is the predicted future state of the AV at time $t+\tau$ from trajectory $\mathcal{T}^e_t$ planned at time $t$ using the goal $\pi(s_{t,0})$, $s^{o,k}_{t,\tau}$ is the predicted future state of the $k$th other traffic participant at time $t+\tau$ from prediction $\mathcal{T}^{o,k}_t$ predicted at time $t$, $s^o_{t,\tau}$ is the predicted future states of $n$ other traffic participants at time $t+\tau$ from prediction at time $t$. In addition, $f$ is a function that outputs the state $s_{t,\tau}$ for the RL agent by combining the predicted future states of the AV $s^e_{t,\tau}$ and the other traffic participants $s^o_{t,\tau}$, $\delta$ is a sufficiently small time step size, $g$ is a function that predicts the reward $r_{t,\tau+\delta}$ at time $t+\tau+\delta$ during transition from $s_{t,\tau}$ to $s_{t,\tau+\delta}$, $L(\theta^Q)$ is the loss for the action value function $Q^\pi$, and $y_t$ is the expected return inferred by utilizing predicted rewards and the action value of the predicted state at the planning horizon, and its corresponding action. Fig.2 shows the predicted states of the other vehicle and the AV with its goal at time $t$.

Although the RP stabilizes learning by utilizing the expectations of future states, its variance is not considered; however, in terms of predicting the states of other traffic participants and the AV, its variance is strongly related to safety. Thus, we utilize the uncertainty propagation on future states of other traffic participants and the AV to consider the variance. Additional details about the uncertainty propagation process are given in Section III-D.

### C. Iterative Reward Prediction

Note that the rewards predicted by the RP still is inaccurate and has errors from the true rewards. Even though the AV can track the given trajectory perfectly, the planned trajectory in the next time step will differ slightly from the previously planned trajectory, which causes inaccurate prediction of reward. In turn, inaccurate prediction of reward results in learning instability and poor performance. Fig.3 demonstrates the inaccuracy problem of RP.

To resolve the inaccuracy problem of RP, we propose the IRP method, which predicts the reward by iteratively planning a new trajectory at the predicted state and predicting the reward of that trajectory. Fig. 4 shows how the proposed IRP method operates. Compared to the conventional RP, the proposed IRP predicts the reward more accurately and

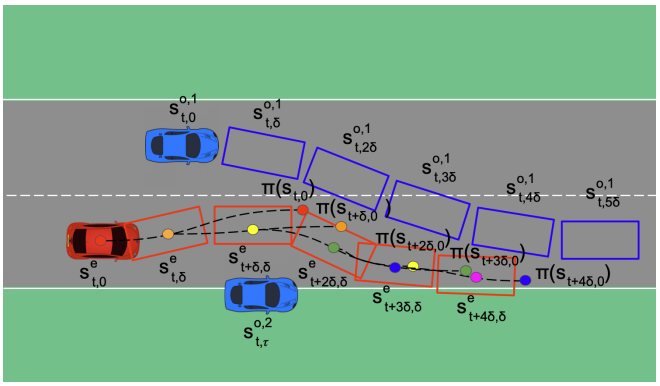


Fig. 4: Predicted states(blue boxes) of other traffic participants(blue cars) and predicted states(red boxes) of the AV(red car) and actions of the RL agent. The action of the RL agent is determined iteratively from the previous state of the AV, its goal, and the states of the other vehicles. The goals and the AV's start state are represented as red, orange, yellow, green, blue, and purple circles in chronological order. The predicted states of the other vehicles and the iteratively predicted states of the AV are utilized to predict the reward during the learning process.

reduces the error of the function approximation of the action value function $Q^\pi$. The mathematical representation of the IRP is given as follows:

$$
\begin{aligned}
&s_{t+\tau,\delta} = h(s_{t+\tau,0}, \pi(s_{t+\tau,0})),\\
&s_{t+\tau+\delta,0} = s_{t+\tau,\delta},\\
&r_{t+\tau,\delta} = g(s_{t+\tau,0}, s_{t+\tau,\delta}),\\
&J(\pi) = Q^\pi(s_t, a_t),\\
&L(\theta^Q) = \mathbb{E}_{s_t\sim\rho^\pi, a_t\sim\pi}[(Q^\pi(s_t,a_t) - y_t)^2],\\
&y_t = r_{t,\delta} + \gamma \cdot r_{t+\delta,\delta} + \cdots + \gamma^{T/\delta-1} \cdot r_{t+T-\delta,\delta}\\
&\qquad + \gamma^{T/\delta} \cdot Q^\pi(s_{t+T-\delta,\delta}, \pi(s_{t+T-\delta,\delta})),
\end{aligned}
\tag{5}
$$

where time $t$ is when planning first started, $s_{t+\tau,\delta}$ is the predicted state at time $t+\tau+\delta$ predicted from the previously predicted state at time $t+\tau$, and $h$ is a function that outputs a state $s_{t+\tau,\delta}$. Here, the predicted states $s_{t+\tau,\delta} = s_{t+\tau+\delta,0}$ are used iteratively to predict the next states $s_{t+\tau+\delta,\delta}$. In function $h$, the future states of the other traffic participants are assumed to follow the trajectories predicted at time $t$, which is assumed to be given. However, the future states of the AV are updated iteratively at each time step.

In addition, considering the agent's future action more accurately enables the agent to drive along a complicated trajectory and stabilizes the learning process because the agent has more comprehensive understanding of its own future actions. As shown in Fig.4, the AV can plan a red trajectory by considering agent's future actions; thus, it is capable of driving a complicated trajectory.

### D. Uncertainty Propagation

During RP and IRP, the expected future states are considered; however, the variance and uncertainty of the predicted future states are not considered. Note that not considering the variance of the prediction leads to inaccurate function approximation and the RL agent being unaware of the uncertainty. In addition, ignoring the uncertainty can cause

the agent to perform sudden deceleration or become involved in an accident. For example, if a low-functioning controller is employed to track a trajectory, and the RL agent does not take that into account, then the AV may encounter various dangerous situations and the predicted rewards may have significant error. To consider the variance of the prediction and the uncertainty of future states, we utilize uncertainty propagation on the RP and IRP motivated by [21], [22].

The uncertainty propagation process is built on the Kalman filter; however, the measurement update process of the Kalman filter is removed because observing future states is impossible. The uncertainty propagation utilized on the RP is expressed as follows:

$$\begin{aligned} &\check{s}^e_{t,\tau} \sim N(s^e_{t,\tau}, \Sigma^e_{t,\tau}),\ \check{s}^o_{t,\tau} \sim N(s^o_{t,\tau}, \Sigma^o_{t,\tau}) \\ &\Sigma^e_{t,\tau+1} = F\Sigma^e_{t,\tau}F^T + Q^e_\tau,\ \Sigma^o_{t,\tau+1} = F\Sigma^o_{t,\tau}F^T + Q^o_\tau, \end{aligned} \tag{6}$$

where $\Sigma^e_{t,\tau}$, and $\Sigma^o_{t,\tau}$ represent the covariance of the Gaussian random variables $\check{s}^e_{t,\tau}$, and $\check{s}^o_{t,\tau}$, respectively, $F$ is the state transition matrix and $Q^e_\tau$, $Q^o_\tau$ represent the process noise of the AV and the other traffic participants, respectively, where $Q^e_\tau$ is attributed to localization, control error, and $Q^o_\tau$ is attributed to object detection, trajectory prediction error respectively. In addition, $\check{s}^e_{t,\tau}$ and $\check{s}^o_{t,\tau}$ are used in IRP to perform collision checking. Here, the ellipse defined by the covariance matrix can provide an upper bound of the probability $1-\delta$ that the AV and other traffic participants exist. However, to check for a collision between the AV and other traffic participants, the rectangle shapes of the AV and other traffic participants must be considered. In the proposed method, We compute the Minkowski sum of the rectangle and the ellipse [21]. The new shape from the Minkowski sum is then utilized for collision checking, which guarantees probability of $(1-\delta)^2$ whether collide or not. The uncertainty propagation and collision checking process using the Minkowski sum are illustrated in Fig.5

### E. Overall Algorithm

The proposed method is based on existing deterministic policy gradient algorithms [20], [26]. We modified the critic update process by utilizing the IRP, and the uncertainty propagation method. The pseudocode of the proposed method is given in Algorithm 1. Here, in lines 6-7, the uncertainty propagation and Minkowski sum are executed using the state of the AV and the other traffic participants $s^e_{t+\tau,0}, s^o_{t+\tau,0}$. In line 9, the state $s_{t+\tau,\delta}$ is predicted by the agent's predicted action $\pi(s_{t+\tau,0})$, where $s_{t+\tau,0}$ comprises the predicted AV's state $s^e_{t+\tau-\delta,\delta}$ and the other traffic participants' predicted state $s^o_{t,\tau+\delta}$ from the trajectory prediction module at time $t$. In line 10, the reward is predicted during the transition from $s_{t+\tau,0}$ to $s_{t+\tau,\delta}$.

## IV. Experiments

The proposed method and baseline methods were evaluated using the CARLA simulator. The experimental configuration, baseline methods, and implementation details are described in the following.

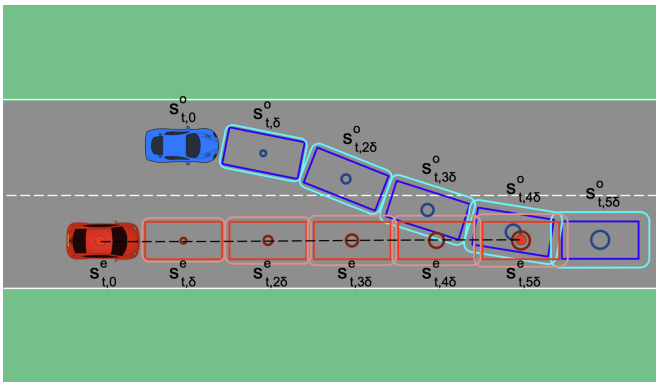


Fig. 5: Predicted states of the AV(red) and another traffic participant(blue). The state of the AV and the goal are represented as a red circles. The predicted states of the other traffic participant and the AV are represented as blue boxes and red boxes for, respectively. Their covariance is represented as the dark blue and dark red circles in the center of the corresponding boxes, respectively. Note that a larger covariance results in a larger circle. The Minkowski sum of each state and the covariance are shown in light blue and light red rounded rectangle, respectively.

**Algorithm 1** Pseudo code of proposed method

1: **procedure** CRITICUPDATE( )
2: $s^e \leftarrow s^e_{t,0}, s^o \leftarrow s^o_{t,0}$
3: $r' \leftarrow r_{t,\delta}$ ▷ Initialize predicted reward with received reward from the simulator
4: $\tau \leftarrow \delta$
5: **while** $\tau < T$ **do**
6: $\check{s}^e, \check{s}^o \leftarrow UncertaintyPropagation(s^e, s^o)$
7: $s^{e\prime}, s^{o\prime} \leftarrow MinkowskiSum(\check{s}^e, \check{s}^o)$
8: $s \leftarrow f(s^{e\prime}, s^{o\prime})$ ▷ Merge States
9: $s' \leftarrow h(s, \pi(s))$ ▷ Prediction of $s_{t+\tau,\delta}$
10: $r \leftarrow g(s, s')$ ▷ Prediction of $r_{t+\tau,\delta}$
11: $r' \leftarrow r' + \gamma^{\tau/\delta} r$ ▷ Update Predicted Reward
12: $s^e, s^o \leftarrow f^{-1}(s')$ ▷ Update Next State
13: $\tau \leftarrow \tau + \delta$
14: **end while**
15: $\check{s}^e, \check{s}^o \leftarrow UncertaintyPropagation(s^e, s^o)$
16: $s^{e\prime}, s^{o\prime} \leftarrow MinkowskiSum(\check{s}^e, \check{s}^o)$
17: $s' \leftarrow f(s^{e\prime}, s^{o\prime})$ ▷ Merge States
18: Set $y_t = r' + \gamma^{T/\delta} Q(s', \pi(s'|\theta^\pi)|\theta^Q)$
19: Update critic by minimizing the loss:
20: $L(\theta^Q) = (Q(s_t, a_t|\theta^Q)) - y_t)^2$
21: **end procedure**

### A. Experiment Configuration

We evaluated the proposed and baseline methods in four distinct scenarios. Scenario 1 involved a lane following scenario with static obstacles, scenario 2 involved a lane following scenario with traffic participants, scenario 3 involved a lane changing scenario with traffic participants, and scenario 4 is an overtaking parked cars scenario with traffic participants. All necessary inputs, e.g., route path, object detection, trajectory prediction, and localization were given. Success was determined if the AV reached the goal without a collision within specified time. In this evaluation, the goal was $130m$ ahead of the AV's initial position. Here,

a maximum lateral deviation of $1.5m$ from the center of the target lane was permitted.

In each scenario, the AV was spawned on the road with a random lateral deviation of $[-1.5m, 1.5m]$ from the center of the road, and a random heading angle deviation of $[-20deg, 20deg]$ with random initial speed of $[5km/h, 15km/h]$. In scenario 1, a maximum of two static obstacles(i.e., vehicles) were spawned at random positions with a lateral deviation of $[-0.5m, 0.5m]$ from the center of the road, and a heading angle deviation of $[-20deg, 20deg]$. Scenario 2 included a maximum of five randomly spawned traffic participants. In scenario 3, the other traffic participants were the same as in scenario 2; however, the AV's goal was to change lanes. In scenario 4, a maximum of two static obstacles(parked cars) and three traffic participants were spawned, and the goal of the AV was to overtake the parked cars while avoiding collisions with the other traffic participants. Note that the traffic participants in each scenario were designed to change lanes randomly.

### B. Baseline Methods

The following two baseline methods were considered in this evaluation. In baseline 1, the output of an agent was a control command, and the input feature was the same as the proposed method. In baseline 2, the output of the agent was the goal of a trajectory which was the same as the proposed method without RP, IRP, and uncertainty propagation. In addition, we evaluated our methods individually as follows: RP, IRP, and IRP with uncertainty propagation.

### C. Implementation Details

All five methods, i.e., baseline1, baseline2, RP, IRP, and IRP with uncertainty propagation were implemented using the DDPG algorithm.

*1) State:* The features of the state space $s$ are composed of $s^e$ and $s^o$. Here, $s^e$ is the state of the AV and comprises $(d, \dot{d}, \ddot{d}, \dot{\sigma}, \ddot{\sigma}, \theta, v_{speed\_limit})$, where $(\sigma, d)$ is the longitudinal and lateral position on the Frenet frame, $\theta$ is the heading angle difference with the center of the road, and $v_{speed\_limit}$ is the speed limit of the road. $s^o$ is composed of $s^{o,k}, k \in \mathbb{N}$, where $\mathbb{N}$ is a natural number. $s^{o,k}$ is the state of the $k$th traffic participant, which is composed of $(\sigma, d, \theta', l, w, v_\sigma, v_d)$, where $(\sigma, d)$ is the position, $\theta'$ is the heading angle difference, $(v_\sigma, v_d)$ is the velocity of the $k$th traffic participant on the Frenet frame, and $(l, w)$ represents the length and width of the vehicle in consideration of the Minkowski sum.

*2) Action:* The action space $a$ is the goal of the trajectory which is composed of $(T_{target}, d_{target}, \sigma_{target}, \dot{\sigma}_{target})$, where $T_{target}$ is the time interval between the current state and the goal state, and $\sigma_{target}, d_{target}$ are the longitudinal and lateral target positions in the Frenet frame, and $\dot{\sigma}_{target}$ is the target longitudinal speed. Here, the trajectory planning method [28] is employed to plan a trajectory toward the goal. The lateral jerk-optimal trajectory is generated given the initial state of the AV $[d, \dot{d}, \ddot{d}]$, and the end state $[d_{target}, \dot{d}_{target} = 0, \ddot{d}_{target} = 0]$ at $T_{target}$ from the action. In addition, the longitudinal jerk-optimal trajectory is generated given the initial state of the AV $[\sigma, \dot{\sigma}, \ddot{\sigma}]$, and the end state $[\sigma_{target}, \dot{\sigma}_{target}, \ddot{\sigma}_{target} = 0]$ at $T_{target}$ from the action. The final trajectory is obtained by combining the lateral and longitudinal trajectories. Note that the planned trajectory is tracked using an MPC-based controller.

*3) Reward:* The reward function is designed to encourage safe, comfortable, and efficient driving as follows:

$$\begin{aligned} r =& \lambda_{lat\_acc} \cdot a_{lat\_acc} + \lambda_{lat\_jerk} \cdot a_{lat\_jerk} \\ &+ \lambda_{long\_acc} \cdot a_{long\_acc} + \lambda_{long\_jerk} \cdot a_{long\_jerk} \\ &+ \lambda_d \cdot |d| + \lambda_v \cdot |v - v_{des}| + r_{collision}, \end{aligned} \tag{7}$$

where $\lambda_{lat\_acc}$, and $a_{lat\_acc}$ represent the weight and penalty for the lateral acceleration, $\lambda_{lat\_jerk}$, and $a_{lat\_jerk}$ are the weight and penalty for the lateral jerk, $\lambda_{long\_acc}$, and $a_{long\_acc}$ are the weight and penalty for the longitudinal acceleration, $\lambda_{long\_jerk}$, and $a_{long\_jerk}$ are the weight and penalty for the longitudinal jerk, $\lambda_d$, and $|d|$ are the weight and penalty for the lateral deviation from the target lane, $\lambda_v$, and $|v - v_{des}|$ are are the weight and the penalty for being slower or faster than the desired speed respectively, and $r_{collision}$ represents the reward and penalty for a collision event. Here, $r_{collision}$ is negative when a collision occurs and positive when no collision occurs. Note that the above reward is also utilized during RP.

## V. Results

Fig. 6, shows the average reward per time step in an episode, the collision rate, and the success rate of the compared methods for all scenarios during training. Table. I, shows the best scores of each of the compared methods for all scenarios during training. In Fig. 6, baseline 2 demonstrates greater learning stability than baseline 1 in every scenario. In terms of performance, in scenario 1, baseline 2 exhibits a $14.49\%$ lower collision rate and a $50.30\%$ higher success rate compared to baseline 1, in scenario 3, baseline 2 achieves an average reward of $0.1424$ that is $19.5$ times higher than that of baseline 1. Baseline 2 is more robust against action variance compared to baseline1, which enables baseline 2 to explore and identify optimal states and actions, leading to stable learning and improved performance. In Fig. 6, the proposed methods(i.e., the agent with RP, IRP, and IRP with uncertainty propagation) demonstrates stable learning compared to the baseline methods. In terms of performance, the proposed methods outperformed the baseline methods in all scenarios, with their performance improving sequentially. In scenario 2, the average reward of the RL agent with RP was $0.2200$, which was $5.7$ times higher than the baseline 1 and $2.4$ times higher than the baseline 2, the collision rate was $16.75\%$, which was $44.96\%$ lower than the baseline 1 and $26.29\%$ lower than the baseline2, and the success rate was $82.79\%$, which was $61.11\%$ higher than the baseline 1 and $26.06\%$ higher than the baseline 2. This increase in performance is attributed to the reduction of function approximation error by the RP method. Additionally, in scenario 4, the average reward of the RL agent with IRP was $0.2173$, which was $11.62\%$ higher than the agent with

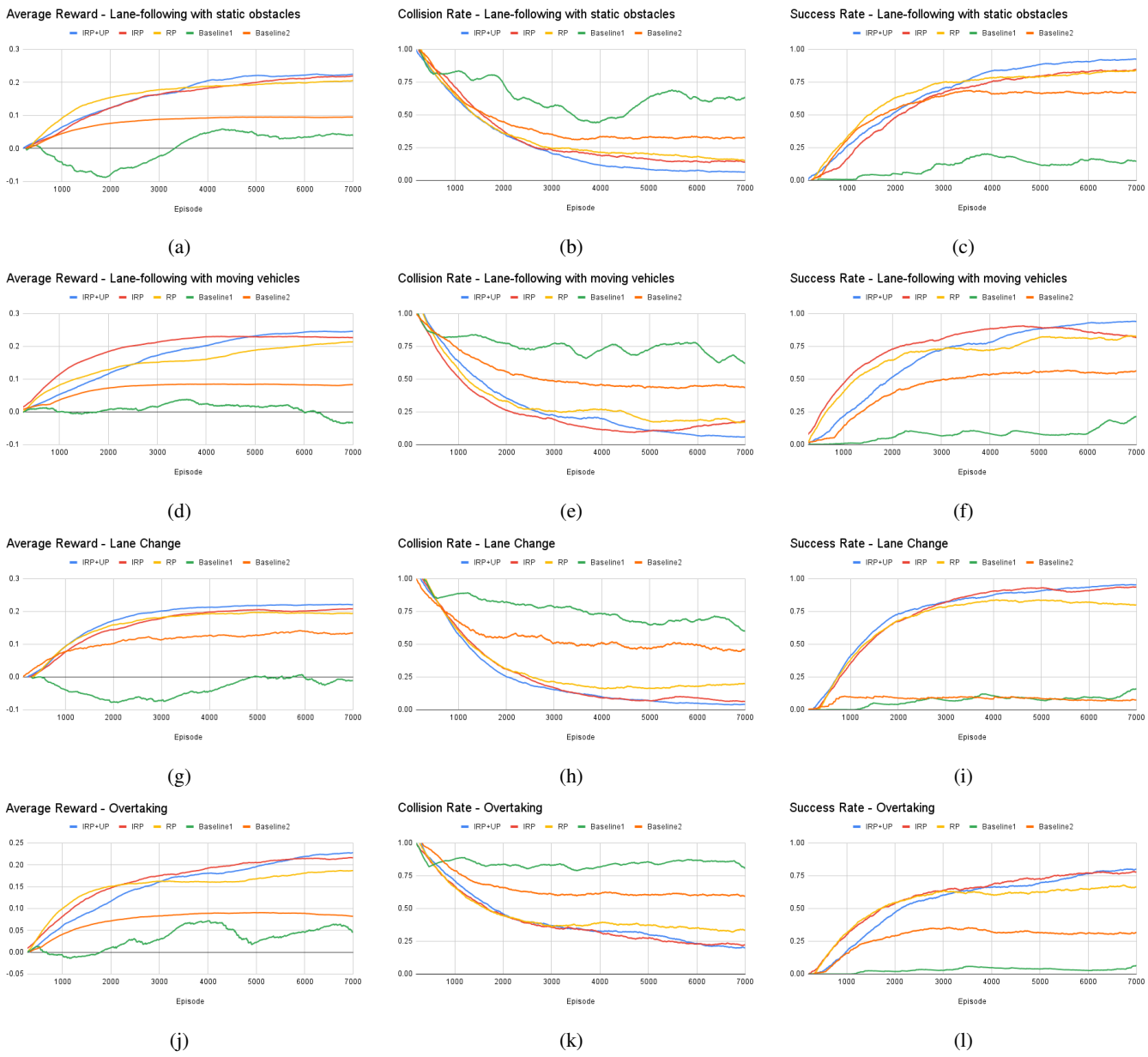


Fig. 6: Average reward per time step in an episode, collision rate, success rate of all methods in every scenario during training.

RP. The collision rate was 21.55%, which was 10.32% lower than the agent with RP, and the success rate was 78.40%, which was 10.73% higher than the agent with RP. The RL agent with IRP outperformed the agent with RP because IRP method predicts the agent's action more accurately, leading to more precise reward predictions and better overall performance. The RL agent with IRP and uncertainty propagation performed even better than the agent with IRP due to the consideration of uncertainty. In scenario 1, the average reward of the RL agent with IRP and uncertainty propagation was 0.2297, which was 4.69% higher than the agent with IRP. The collision rate was 5.682%, which was 7.788% lower than the agent with IRP, and the success rate was 93.86%, which was 9.05% higher than the agent with IRP.

In scenario 3, even though lane changes are quite challenging in typical road situations, the collision rate was relatively small compared to the other scenarios. The reason for this is that the other traffic participants used in the experiment were designed to drive conservatively. The conservative driving of the other traffic participants reduced the collision rate. For scenario 4, the collision rate was much higher than in scenarios 1-3 because avoiding parked cars among the other traffic participants was considerably more challenging compared to the other scenarios. A video of the agent with IRP and uncertainty propagation method is available at `https://www.youtube.com/watch?v=PfDbaeLfcN4`.

TABLE I: Best results among the compared methods

| Scenario | | Average reward | Collision rate | Success rate |
|---|---|---|---|---|
| 1 | Baseline 1 | 0.0589 | 42.81% | 20.65% |
| | Baseline 2 | 0.0942 | 28.32% | 70.95% |
| | RP | 0.1854 | 14.83% | 84.15% |
| | IRP | 0.2194 | 13.47% | 84.81% |
| | IRP+UP | **0.2297** | **5.682%** | **93.86%** |
| 2 | Baseline 1 | 0.0383 | 61.71% | 21.68% |
| | Baseline 2 | 0.0912 | 43.04% | 56.73% |
| | RP | 0.2200 | 16.75% | 82.79% |
| | IRP | 0.2252 | 8.385% | 91.50% |
| | IRP+UP | **0.2460** | **5.782%** | **94.11%** |
| 3 | Baseline 1 | 0.0073 | 60.04% | 15.81% |
| | Baseline 2 | 0.1424 | 42.12% | 19.62% |
| | RP | 0.1980 | 14.16% | 85.86% |
| | IRP | 0.2083 | 5.913% | 93.85% |
| | IRP+UP | **0.2250** | **3.590%** | **96.08%** |
| 4 | Baseline 1 | 0.0694 | 78.43% | 6.911% |
| | Baseline 2 | 0.0909 | 54.65% | 37.84% |
| | RP | 0.1870 | 31.87% | 67.67% |
| | IRP | 0.2173 | 21.55% | 78.40% |
| | IRP+UP | **0.2299** | **18.26%** | **81.91%** |

## VI. Conclusions And Future Work

In this study, we have proposed a method that includes RP, IRP, and uncertainty propagation methods to reduce the function approximation error of an AV's RL-based planning agent. The proposed method was evaluated under several scenarios, and the results demonstrated that the proposed method improves both learning stability and agent performance compared to baseline methods. However, result of the evaluation showed that the proposed method still has poor performance in difficult and complex scenarios. Future works will involve increasing the safety while having better performance.

## References


[1] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

[2] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

[3] Stefano Pini, Christian S Perone, Aayush Ahuja, Ana Sofia Rufino Ferreira, Moritz Niendorf, and Sergey Zagoruyko. Safe real-world autonomous driving by learning to predict and plan with a mixture of experts. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10069–10075. IEEE, 2023.

[4] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[5] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[6] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE intelligent transportation systems conference (ITSC)*, pages 2765–2771. IEEE, 2019.

[7] Jianyu Chen, Shengbo Eben Li, and Masayoshi Tomizuka. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5068–5078, 2021.

[8] Dhruv Mauria Saxena, Sangjae Bae, Alireza Nakhaei, Kikuo Fujimura, and Maxim Likhachev. Driving in dense traffic with model-free reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5385–5392. IEEE, 2020.

[9] Guanlin Wu, Wenqi Fang, Ji Wang, Pin Ge, Jiang Cao, Yang Ping, and Peng Gou. Dyna-ppo reinforcement learning with gaussian process for the continuous action decision-making in autonomous driving. *Applied Intelligence*, 53(13):16893–16907, 2023.

[10] Błażej Osiński, Adam Jakubowski, Paweł Zięcina, Piotr Miłoś, Christopher Galias, Silviu Homoceanu, and Henryk Michalewski. Simulation-based reinforcement learning for real-world autonomous driving. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 6411–6418. IEEE, 2020.

[11] Lingping Gao, Ziqing Gu, Cong Qiu, Lanxin Lei, Shengbo Eben Li, Sifa Zheng, Wei Jing, and Junbo Chen. Cola-hrl: Continuous-lattice hierarchical reinforcement learning for autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13143–13150. IEEE, 2022.

[12] Ziqing Gu, Lingping Gao, Haitong Ma, Shengbo Eben Li, Sifa Zheng, Wei Jing, and Junbo Chen. Safe-state enhancement method for autonomous driving via direct hierarchical reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[13] Zhiqian Qiao, Jeff Schneider, and John M Dolan. Behavior planning at urban intersections through hierarchical reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2667–2673. IEEE, 2021.

[14] Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022.

[15] Xiaobai Ma, Jiachen Li, Mykel J Kochenderfer, David Isele, and Kikuo Fujimura. Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6064–6071. IEEE, 2021.

[16] Tung Phan-Minh, Forbes Howington, Ting-Sheng Chu, Momchil S Tomov, Robert E Beaudoin, Sang Uk Lee, Nanxiang Li, Caglayan Dicle, Samuel Findler, Francisco Suarez-Ruiz, et al. Driveirl: Drive in real life with inverse reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1544–1550. IEEE, 2023.

[17] Yantao Tian, Xuanhao Cao, Kai Huang, Cong Fei, Zhu Zheng, and Xuewu Ji. Learning to drive like human beings: A method based on deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6357–6367, 2021.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[19] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[20] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[21] Wenda Xu, Jia Pan, Junqing Wei, and John M Dolan. Motion planning under uncertainty for on-road autonomous driving. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2507–2512. IEEE, 2014.

[22] Jiawei Fu, Xiaotong Zhang, Zhiqiang Jian, Shitao Chen, Jingmin Xin, and Nanning Zheng. Efficient safety-enhanced velocity planning for autonomous driving with chance constraints. *IEEE Robotics and Automation Letters*, 2023.

[23] Tianqi Qie, Weida Wang, Chao Yang, Ying Li, Yuhang Zhang, Wenjie Liu, and Changle Xiang. An improved model predictive control-based trajectory planning method for automated driving vehicles under uncertainty environments. *IEEE Transactions on Intelligent Transportation Systems*, 24(4):3999–4015, 2022.

[24] Constantin Hubmann, Jens Schulz, Marvin Becker, Daniel Althoff, and Christoph Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. *IEEE transactions on intelligent vehicles*, 3(1):5–17, 2018.

[25] Shivesh Khaitan, Qin Lin, and John M Dolan. Safe planning and

control under uncertainty for self-driving. *IEEE Transactions on Vehicular Technology*, 70(10):9826–9837, 2021.

[26] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[28] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE international conference on robotics and automation*, pages 987–993. IEEE, 2010.