2023 IEEE 26th International Conference on
Intelligent Transportation Systems (ITSC)
24-28 September 2023. Bilbao, Bizkaia, Spain

Generation of Adversarial Trajectories using Reinforcement Learning to Test Motion Planning Algorithms

Joshua Ransiek, Barbara Schütt, Adrian Hof and Eric Sax[1]

***Abstract*— Autonomous vehicles must be comprehensively tested before being deployed in the real world. Simulators offer the possibility of safe, low-cost development of self-driving systems. In addition to environmental perception, motion planners are particularly important to provide robust and safe driving trajectories. However, evaluating and improving motion planning algorithms for autonomous vehicles requires scalable generation of traffic scenarios. To be useful, these scenarios must be realistic and challenging, but also remain possible to traverse safely. In this work, we introduce a reinforcement learning (RL) approach for automatically generating adversarial trajectories to challenge motion planning algorithms. Given a motion planner, we train our deep RL agent to search the risky trajectory space of an adversarial vehicle. Actor-critic RL optimizes the entire process. Through experiments conducted on multiple routes within a selected intersection scenario, we demonstrate the effectiveness of our proposed framework in generating adversarial trajectories that minimize state-of-the-art scenario criticality measures.**

***Index Terms*— Reinforcement Learning, Adversarial Generation, Scenario-based Testing, Motion Planning**

I. Introduction

In recent years, the rapid enhancement of Advanced Driver-assistance Systems (ADAS) has transformed the automotive industry, accompanied by consequential changes in the type approval process for driving functions. These systems hold potential for enhancing road safety, improve traffic efficiency, and support of the human driver [1]. However, ensuring their reliability and robustness remains a critical challenge. The use of test scenarios has become mandatory under current European Union (EU) legislation [2]. For specific ADAS, e.g., Lane Keep Assist, a predefined list of scenarios is provided, and the regulation states that a driving function generally must be tested using scenarios derived from its intended Operational Design Domain (ODD). By conducting scenario-based tests, engineers can assess their system's capability to handle complex driving situations, mitigate risks, and make appropriate decisions. Nevertheless, the effectiveness of scenario-based testing heavily relies on the quality and diversity of the testing scenarios. Moreover, with growing automation, the needed test effort is increasing as well [3].

Current practices for scenario generation in the scenario-based testing predominantly rely on engineers or involve a semi-automated process. In the semi-automated approach, a human operator specifies the number of actors, their initial positions, and predetermined trajectories, while their speeds

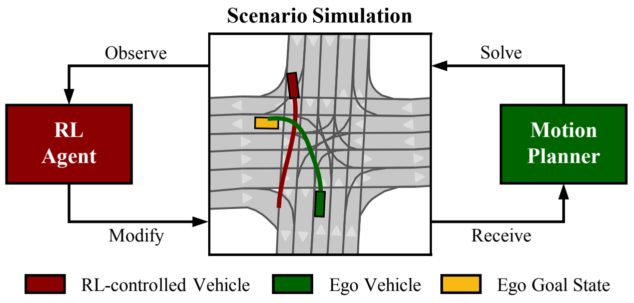

Fig. 1. Our pipeline for adversarial trajectory generation. For a scenario from the CommonRoad repository, we hijack one vehicle (red), the target vehicle, for adversarial movement. After observing the scenario, a RL agent modifies the target vehicle's trajectory, while a motion planner tries to solve the modified scenario by planning a path for the ego vehicle (green) to reach the planning goal region (yellow). The RL agent improves the target vehicle's trajectory towards the minimum of a scenario criticality measure.

are selected from a range of options. However, human involvement can be time-consuming and may lead to critical scenarios being overlooked. To address this challenge, researchers are actively exploring the use of agents in simulations to create collisions or exhibit adversarial behavior. However, these agents often require training to maneuver vehicles effectively, and their generated scenarios may surpass the system's capabilities, resulting in unrealistic situations. In this paper, we propose a comprehensive approach to scenario generation for testing of automated driving system (ADS), specifically motion planning algorithms. Our approach addresses the limitations of existing methods and offers the following key contributions:

1) **Scalable and Automated Process:** The proposed RL-based methodology automatically generates adversarial trajectory, scales to different routes, and bypasses the necessity for hand-crafted rules and heuristics. Nevertheless, strict requirements and human input can be considered through the design of the reward function.
2) **Trajectory Optimization:** To generate diverse and realistic trajectories, the approach simultaneously optimizes the initial position, lateral movement, and velocity of the adversarial vehicle based on the performance of the motion planner.
3) **Generation on Scenario-level:** Our approach controls the adversarial vehicle's trajectory at the scenario level, without the need for the vehicle to learn individual control. Each step in the RL framework corresponds to a scenario simulation, allowing for effective generation of challenging scenarios.

In Section II, we provide a concise overview of the rele-

[1]Joshua Ransiek, Barbara Schütt, Adrian Hof and Eric Sax are with the FZI Research Center for Information Technology, 76131 Karlsruhe, Germany, {ransiek, schütt, sax}@fzi.de

vant literature on scenario generation, simulators, and the utilization of RL in this context. The proposed approach for adversarial trajectory generation is elaborated in detail in Section III, outlining the methodology and algorithms employed. Subsequently, in Section IV, we present the experimental results obtained through the application of our approach. We showcase the efficacy of our approach in generating challenging and realistic scenarios that push the boundaries of ADS and ADAS capabilities. Finally, in Section V, we provide a summary of our findings, emphasize the significance of our work and identify potential directions for future research and development in this domain.

II. Related Work

A. Safety-Critical Scenario Generation

The development of ADSs necessitates the creation of diverse and, ideally, realistic scenarios to ensure safe and efficient operation of the system in various traffic situations and ODDs. Schütt *et al.* [4] classify the acquisition of new scenarios into different categories, including scenario alteration, scenario exploration, and scenario generation. Scenario alteration and exploration aim to identify new scenarios by modifying scenario parameters with new and improved values to meet specific goals, such as identifying critical or near-crash scenarios. For instance, [5] employed an evolutionary algorithm to alter actor trajectories, or [6] utilized Bayes optimization with Gaussian Processes to identify the most critical scenarios within a logical scenario.

On the other hand, scenario generation involves the creation of entirely new scenarios, such as novel sequences of actor states or actor behavior. The primary objective is not only to identify scenarios associated with the initial scenario or scenario set but also to discover unrelated ones, preferably. To achieve this goal, Goss *et al.* [7] have proposed a modular scenario framework where basic logical scenarios extracted from accident data are stored as atomic blocks/scenarios. These atomic blocks can be combined in the second step to obtain new, more challenging scenarios that differ from existing ones. In another example, Paranjape *et al.* [8] suggest a procedural approach that generates maps and agents (vehicles and pedestrians) on-the-fly. The agents are governed by behavior trees, and the modular behavior concept enables the creation and testing of a broader range of scenarios throughout the simulation.

Criticality metrics serve as essential tool to evaluate generated scenarios. However, the majority of these metrics operate in a binary manner, focusing on the criticality between two specific traffic participants. While metrics such as Euclidean distance between vehicles offer simplicity in their calculation, their reliability in assessing criticality may be compromised in certain scenarios, particularly when participants drive alongside each other in the same direction. Another well-known metric, Time-To-Collision (TTC) [9], considers the distance and velocities of leading and following vehicles. However, its applicability is restricted to car-following situations and may not be applicable in other scenarios. To address this limitation, several approaches for TTC-related metrics have been proposed to extend their usability beyond limited domains. One such approach is the Worst-Time-To-Collision (WTTC) [10], which evaluates the criticality between two actors by assuming the most unfavorable intersecting trajectories they could choose at maximum speed.

B. Simulators

Simulators offer a powerful means to train, test and validate ADSs on a large scale all while avoiding public safety concerns. The credibility and fidelity of the virtual simulation have a profound impact on the system being tested. High-fidelity, photo-realistic simulators provide data that is comparable to the physical world, enabling a more accurate assessment of ADSs performance. They allow for the simulation of realistic lighting and weather conditions, including variables such as precipitation strength, cloudiness, and fog density. These capabilities make them well-suited for evaluating perceptual functions and overall system behavior. CARLA [11] is a high-fidelity 3D simulator for the development, training, and performance analysis of autonomous driving systems. It leverages semantic segmentation to estimate lanes, road boundaries, and dynamic objects. ADSs can be tested by navigating through urban environments alongside other vehicles and pedestrians. SUMMIT [12] generates high-fidelity, interactive data for unregulated, dense urban traffic on real-world maps. SUMMIT supports the development and testing of driving algorithms.

On the other hand, low-fidelity simulators are unable to support the detailed and photo-realistic simulations offered by their high-fidelity counterparts. However, they possess advantages such as being lightweight and user-friendly, allowing for quick testing of algorithms without requiring extensive configuration. These simulators typically assume ideal perception and are primarily utilized for training and evaluating motion planning and control algorithms. CommonRoad [13] is a low-fidelity 2D simulator for motion planning that allows evaluation and comparison of different types of motion planners. By utilizing sets of object trajectories and road points, motion planners navigate towards their planning goals while ensuring safety. Nocturne [14] is another low-fidelity 2D driving simulator for investigating multi-agent coordination. It operates without computer vision and avoids feature extraction from images. RL agents within Nocture observe vector representations of object sets and road points, mimicking the perspective of an idealized human driver. MetaDrive [15] is a compositional 3D driving simulator that offers an extensive range of driving scenarios. It achieves this through procedural generation and replay of real traffic data. Further, MetaDrive trades visual quality for high sample efficiency and extensibility, enabling support for various research topics such as safe RL and multi-agent autonomy.

C. Reinforcement Learning

RL addresses the problem of an agent learning to act in an environment defined as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$. The MDP consists of a state space $\mathcal{S} \subset \mathbb{R}^n$, action space $\mathcal{A} \subset \mathbb{R}^m$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, transition dynamics p, and discount factor $\gamma \in [0, 1)$ [16]. At each discrete time step t, the environment provides the agent with a state $\boldsymbol{s}_t$, the agent responds by selecting an action $\boldsymbol{a}_t$, and then the environment provides the next reward $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and state $\boldsymbol{s}_{t+1}$. For convenience, we use the simpler notations of r, $\boldsymbol{s}$, $\boldsymbol{a}$, $\boldsymbol{s}'$, and $\boldsymbol{a}'$ to refer to a reward, state, action, next state, and next action, respectively. The ultimate objective of the RL agent is to find its policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected discounted cumulative return:

$$\underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]. \qquad (1)$$

Traditional RL methods have shown promise in solving complex decision-making tasks such as quadrupedal robots [17] and nonlinear parameter estimation [18]. Within the context of ADS testing Baumann *et al.* [19] and Ding *et al.* [20] propose the use of RL to dynamically adjust the initial parameters of actors involved in overtaking and intersection scenarios, thereby identifying critical test cases efficiently compared to grid search, random sampling, and human-designed methods. Koren *et al.* [21] introduce adaptive stress testing, a method for testing the autonomous vehicles' decision making systems. They employ deep RL to identify likely failure scenarios in a crosswalk scenario by controlling the lateral and longitudinal acceleration of pedestrians. Furthermore, Kuutti *et al.* [22] employ an adversarial RL agent in a vehicle following scenario, where the objective is to degrade the performance of the system-under-test by inducing collisions.

III. Generating Adversarial Trajectories

This section introduces our novel deep RL approach for generating adversarial trajectories. The proposed framework is depicted in Fig. 2. We begin by presenting the scenario representation and traffic simulator utilized in our approach. Next, we describe the selected motion planner and RL algorithm. Finally, we outline the trajectory generation pipeline, which encompasses the training routine of the deep RL agent.

A. Simulator

To facilitate the generation and modification of traffic scenarios, we leverage the lightweight design and convenient Python API provided by the CommonRoad framework [13]. CommonRoad offers a versatile set of benchmarks for motion planning on roads that provide the planner with an ideal representation of the traffic scene. Traffic situations in the CommonRoad framework are defined as scenarios encompassing the road geometry and a collection of traffic participants that can either have static or dynamic behavior. The road network is specified by means of lanelets [23]. These are atomic, interconnected and drivable road segments. Each scenario includes a planning problem for the ego vehicle. The primary objective of the ego vehicle is to navigate through the road network within a specified time frame, starting at an initial state, ensuring collision avoidance and adherence to the boundaries of the road network, and ultimately reaching a designated goal region.

B. Motion Planning Algorithm

CommonRoad supports the connection of uninformed and informed search algorithms, enables the training of RL-based motion planners, and facilitates the use of graph neural networks for solving motion planning problems. To perform motion planning of the ego vehicle, we utilize the A* search algorithm [24]. The A* algorithm is a widely used search algorithm that efficiently determines the optimal path from a given start position to a goal region within a graph-based environment. At each node n in the graph, the cost $f(n)$ is computed:

$$f(n) = g(n) + h(n). \qquad (2)$$

Here $g(n)$ represents the cost of reaching node n from the initial node in the graph and $h(n)$ estimates the cost of transition from node n to the final destination by calculating the distance between n and the final goal. Euclidean distance is used to compute the cost to reach the desired goal position.

C. Reinforcement Learning Algorithm

The trajectory generation approach focuses on model-free off-policy actor-critic RL. We employ actor-critic methods consisting of interleaved policy evaluation and improvement. One of the most widely used off-policy actor-critic algorithms is Soft Actor-Critic (SAC) [25], which can be applied to environments with continuous observation and action spaces. SAC utilizes maximum entropy RL [26], which incorporates an entropy bonus into the original objective (1):

$$\underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t \left[r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha \mathcal{H}\left(\pi(\cdot|\boldsymbol{s}_t)\right) \right] \right]. \qquad (3)$$

Here $\mathcal{H}$ represents the entropy, and α is the temperature parameter that balances exploitation and exploration. During policy improvement, SAC fits two critics to estimate the discounted returns of the current training policy starting at state s and action a:

$$J(\theta_i) = \underset{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')\sim\mathcal{D}}{\mathbb{E}}[(Q_{\theta_i}(\boldsymbol{s}, \boldsymbol{a}) - y(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'))^2],\ i = 1, 2, \qquad (4)$$

$$y(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \Big(\min_{j=1,2} Q_{\theta_{targ,j}}(\boldsymbol{s}', \boldsymbol{a}') - \alpha \log \pi_\phi(\boldsymbol{a}'|\boldsymbol{s}') \Big),\ \boldsymbol{a}' \sim \pi_\phi(\cdot|\boldsymbol{s}'), \qquad (5)$$

where $Q_{\theta_{targ,j}}$ represents the target networks with weights that are updated using exponential moving average (EMA). Using the minimum value of both critic target networks helps to avoid overestimation and therefore reduces overestimating actions. Furthermore, the critic networks are regularized using dropout and layer normalization, as suggested in [27] and [17], to improve sample efficiency. This regularization enables the algorithm to take multiple gradient steps on the

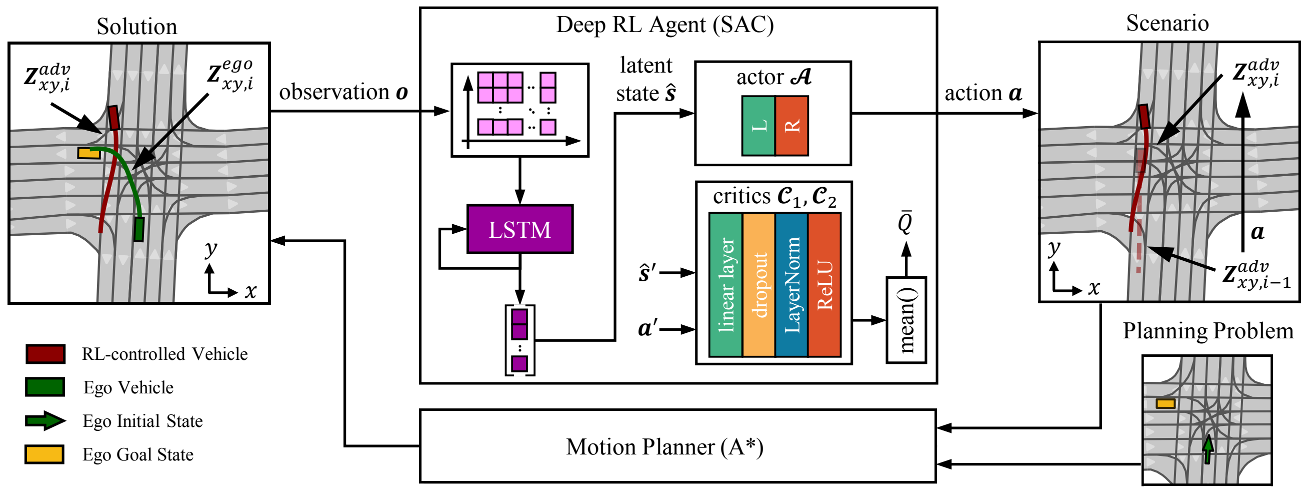

Fig. 2. Structure of the proposed deep RL agent, which observes the trajectories of ego and adversarial vehicle. The extracted feature vector $\hat{\boldsymbol{s}}$ represents the latent state of the environment. The agent's action $\boldsymbol{a}$ incrementally adjust the adversarial vehicle's trajectory $\boldsymbol{Z}^{adv}$. The motion planner plans the safest possible trajectory $\boldsymbol{Z}^{ego}$ of the ego vehicle based on the altered scenario and a predefined planning problem.

critic after each environment step, leading to more sample-efficient learning. According to [27], the critics are used to improve the policy by maximizing the objective:

$$J(\phi) = \mathop{\mathbb{E}}_{\substack{\boldsymbol{s}\sim\mathcal{D} \\ \boldsymbol{a}\sim\pi_\phi(\cdot|\boldsymbol{s})}} \left[\frac{1}{2}\sum_{i=1}^{2} \Big[Q_{\theta_i}(\boldsymbol{s},\boldsymbol{a}) \Big] - \alpha \log \pi_\phi(\boldsymbol{a}|\boldsymbol{s}) \right]. \tag{6}$$

D. Trajectory Generation Framework

Our deep RL approach aims to create demanding traffic situations to challenge a given motion planning algorithm. These situations can then be used to evaluate the effectiveness of the planner. In our approach, the planner encompasses prediction, planning, and control. We are particularly interested in scenarios where the system must intervene and deviate from its comfortable trajectory. Within the CommonRoad [13] framework, the planner receives the road network, other agent's trajectories, and the planning goal as input. It then determines the best possible trajectory for the ego vehicle (the vehicle under control). The planner is considered a black-box, and our deep RL agent has no access to its internal workings. Our main focus is on generating scenarios with a high potential for accidents, achieved by minimizing criticality metrics associated with these scenarios.

The task of generating adversarial trajectories is defined as follows: Agent trajectory i is in the state $\boldsymbol{z}^{\tau}_{xy,i} = (x^{\tau}_i, y^{\tau}_i)$, consisting of its x and y coordinates at timestep τ in a birds-eye perspective. The complete trajectory i of an actor in Cartesian coordinates is denoted as $\boldsymbol{Z}_{xy,i} = \{\boldsymbol{z}^{\tau}_{xy,i} | \tau = 0, \ldots, \mathcal{T}\}$, where $\mathcal{T}$ represents the total number of timesteps in the scenario. To remove the dependence on road geometry for decision making, the trajectory i is transformed into the Frenet frame [28], where the state $\boldsymbol{z}^{\tau}_{sd,i} = (s^{\tau}_i, d^{\tau}_i)$ at timestep τ consists of a longitudinal coordinate s and lateral coordinate d. Consequently, the trajectory i of an actor in the Frenet frame is denoted as $\boldsymbol{Z}_{sd,i} = \{\boldsymbol{z}^{\tau}_{sd,i} | \tau = 0, \ldots, \mathcal{T}\}$.

The objective of trajectory generation is to provide an adversarial agent that exhibits challenging behavior in the given scenario. We approach this problem by formulating it as an optimization problem in the framework of RL, where trajectories are modified by altering the states of the adversarial vehicle in the Frenet frame.

The RL environment, denoted as $\mathcal{E}$, encompasses the motion planner, scenario, and planning problem. The interaction between our deep RL agent and the environment $\mathcal{E}$, along with the detailed structure of the agent is illustrated in Fig. 2. When the agent interacts with $\mathcal{E}$, it receives observations $\boldsymbol{o}$ that provide relevant information about the behavior of the actors in the scenario. The RL agent's observation consist of the adversarial trajectory $\boldsymbol{Z}^{adv}_{xy}$, the ego vehicle trajectory $\boldsymbol{Z}^{ego}_{xy}$ and a scenario criticality metric $\boldsymbol{M}$:

$$\begin{aligned} \boldsymbol{o} &= \begin{bmatrix} \boldsymbol{Z}^{adv}_{xy} & \boldsymbol{Z}^{ego}_{xy} & \boldsymbol{M} \end{bmatrix} && (7) \\ &= \begin{bmatrix} x^{0\ldots\mathcal{T}}_{adv} & y^{0\ldots\mathcal{T}}_{adv} & x^{0\ldots\mathcal{T}}_{ego} & y^{0\ldots\mathcal{T}}_{ego} & m^{0\ldots\mathcal{T}} \end{bmatrix}. && (8) \end{aligned}$$

Here, $x_{adv}, y_{adv}, x_{ego}, y_{ego}$ being the x and y coordinates of the adversarial and ego vehicle trajectory starting at time step 0 until the end of the scenario $\mathcal{T}$. Additionally, the observation includes a criticality m calculated every time step. To preprocess the high-dimensional observation $\boldsymbol{o}$ and extract a compressed feature vector $\hat{\boldsymbol{s}}$, a long short-term memory (LSTM) neural network is utilized. The acting policy π (actor $\mathcal{A}$) as well as the critics $\mathcal{C}_{1,2}$, which estimate the state-action values Q, are based on this feature vector.

Given the observation $\boldsymbol{o}$, or more specifically the feature vector $\hat{\boldsymbol{s}}$, and a reference route ω, which is a sequence of logical consecutive lanelets obtained from the scenario, the agent determines the action $\boldsymbol{a}$. Based on the reference route, a curvilinear coordinate system (CCS) is constructed by transforming it into the Frenet frame. The agent's actions allow it to generate an adversarial trajectory $\boldsymbol{Z}^{adv}_{sd}$ within the CCS. The agent incrementally chooses a starting point a^{adv}_s, velocity a_v^{adv}, and lateral displacement $\boldsymbol{a}_d^{adv}$ to define the adversarial trajectory:

$$\boldsymbol{a} = \begin{bmatrix} a_s^{adv} & \boldsymbol{a}_d^{adv} & a_v^{adv} \end{bmatrix}. \tag{9}$$

Since our approach operates on the scenario level, the entire trajectory is modified simultaneously. The starting point and velocity determine the longitudinal movement of the adversarial vehicle, while lateral movement is directly determined by the RL agent through the lateral displacement. We use B-spline geometries (BSGs) [29] to represent and adapt the high-dimensional continuous space of lateral trajectory values d in the Frenet frame. The B-spline curve is a parametric curve $d(l)$ defined by $m+1$ control points (CPs) and Bernstein polynomials as its basis:

$$d(l) = \sum_{j=0}^{m} p_{d,j} B_{j,m}(l), \quad 0 \leq l \leq 1, \tag{10}$$

where m represents the degree, $p_{d,j}$ represents the j-th CP and $B_{j,m}$ represents the Bernstein basis polynomials:

$$B_{j,m}(l) = \binom{m}{j} l^j (1-l)^{m-j}. \tag{11}$$

According to [30] a cubic curve ($m = 3$) is sufficient to fit nearly all different kinds of straight and curved trajectories. The d-values, required for trajectory calculation, can be retrieved in a favorable discretization through point-wise evaluation of the BSGs. The resulting action space to incrementally adjust the adversarial trajectory is a fixed 6-dimensional vector:

$$\boldsymbol{a} = \begin{bmatrix} \Delta s_{adv} & \Delta p_{d,0} & \Delta p_{d,1} & \Delta p_{d,2} & \Delta p_{d,3} & \Delta v_{adv} \end{bmatrix}, \tag{12}$$

where Δs represents changes to the starting point, Δv represents changes to the velocity and $\Delta p_{d,0\ldots3}$ represents changes to the CP's. In summary, the adversarial trajectory in the Frent frame at step i is adjusted as:

$$\boldsymbol{Z}_{sd,i}^{adv} = \boldsymbol{Z}_{sd,i-1}^{adv} + \Delta\boldsymbol{Z}_{sd,i}^{adv}. \tag{13}$$

The trajectory $\boldsymbol{Z}_{sd,i}^{adv}$ is then transformed back into the Cartesian coordinates $\boldsymbol{Z}_{xy,i}^{adv}$ and used to update the movement of the adversarial vehicle. The new scenario configuration is provided to the motion planner to calculate a new ego vehicle trajectory $\boldsymbol{Z}_{xy,i}^{ego}$, which is subsequently observed by the deep RL agent for further decision-making.

For an incremental update of the adversarial trajectory $\boldsymbol{Z}_{xy}^{adv}$, the agent is rewarded based on the following reward function:

$$r(\boldsymbol{o}, \boldsymbol{a}) = -b_1 \cdot r_{\text{metric}} + b_2 \cdot r_{\text{collision}} - b_3 \cdot r_{\boldsymbol{a}}. \tag{14}$$

where the weightings $\{b_1, b_2, b_3\} \in \mathbb{R}^+$ are design choices that allow for the prioritization of specific aspects of the reward function. The first term represents the risk metric r_{metric} and quantifies the criticality of the scenario. It is calculated as the minimum value of the squared criticality metrics $\boldsymbol{M}$:

$$r_{\text{metric}} = \min\left(\boldsymbol{M}^2\right), \tag{15}$$

To incentivize collisions, a bonus reward $r_{\text{collision}}$ is given when the adversarial trajectory $\boldsymbol{Z}_{xy}^{adv}$ and the the ego trajectory $\boldsymbol{Z}_{xy}^{ego}$ intersect:

$$r_{\text{collision}} = \begin{cases} 1 \text{ if} & \left\|\boldsymbol{Z}_{xy}^{adv} - \boldsymbol{Z}_{xy}^{ego}\right\|_2 < \delta_1 \wedge \\ & \left|\operatorname{atan2}\left(\boldsymbol{Z}_{xy}^{adv} - \boldsymbol{Z}_{xy}^{ego}\right)\right| < \delta_2 \\ 0 \text{ else} & \end{cases}, \tag{16}$$

where δ_1 and δ_2 represent the required distance and angle between the trajectories of the actors. Furthermore, the agent's actions are penalized to discourage unnecessary trajectory alterations due to the incremental nature of the action space:

$$r_a = \frac{1}{N} \sum_n |a_n|. \tag{17}$$

In summary, the proposed agent structure allows for generating accident-prone scenarios to challenge motion planning algorithms. The adapted training routine for the trajectory generation task is described in Algorithm 1. It requires a given scenario $\mathcal{S}$ from which our approach extracts a set of training routes $\mathcal{R}_\omega$. At the beginning the experience replay buffer $\mathcal{R}$, the maximum episode steps K, agent's policy π_ψ and Q-function networks Q_θ are initialized. The LSTM feature extractor is implemented as a flatten layer which is shared between the policy network and the Q-networks. The parameters of the LSTM network are optimized during the policy update step in line 17.

Algorithm 1 Training routine for the adversarial trajectory generation

```
1: input scenario $\mathcal{S}$, training routes set $\mathcal{R}_\omega$
2: initialize deep RL agent policy $\pi_\psi$, Q-networks $Q_\theta$, experience replay buffer $\mathcal{D}$, max. episode steps $K$
3: for each episode do
4:     Sample $\omega$ from $\mathcal{R}_\omega$
5:     $\boldsymbol{d_0} \leftarrow$ Evaluate($\boldsymbol{p}_d$)
6:     $\boldsymbol{Z}_{xy}^{adv} \leftarrow$ CalculateTrajectory($\omega, s_0, \boldsymbol{d_0}, v_0$)
7:     $\boldsymbol{o} \leftarrow$ RunScenario($\boldsymbol{Z}_{xy}^{adv}$)
8:     $k \leftarrow 1$
9:     while $k \leq K$ do
10:        $\boldsymbol{a} \leftarrow \pi_\psi(\boldsymbol{o})$
11:        $s, \boldsymbol{p}_d, v \leftarrow$ Apply($\boldsymbol{a}$)
12:        $\boldsymbol{d} \leftarrow$ Evaluate($\boldsymbol{p}_d$)
13:        $\boldsymbol{Z}_{xy}^{adv} \leftarrow$ CalculateTrajectory($\omega, s_0, \boldsymbol{d}, v_0$)
14:        $\boldsymbol{o}' \leftarrow$ RunScenario($\boldsymbol{Z}_{xy}^{adv}$)
15:        $r \leftarrow$ CalculateRewards($\boldsymbol{o}, \boldsymbol{a}, \boldsymbol{o}'$)
16:        Store $(\boldsymbol{o}, \boldsymbol{a}, r, \boldsymbol{o}')$ in $\mathcal{D}$
17:        Update Q-networks $Q_\theta$
18:        Update policy network $\pi_\psi$
19:        if $k \geq K$ then
20:            End episode
21:        end if
22:        $k \leftarrow k + 1$
23:    end while
24: end for
25: output agent policy $\pi_\psi$
```

For each episode, a new training route ω is sampled from the training routes set $\mathcal{R}_\omega$, and a CCS is placed around it. The route ω remains unchanged throughout the episode. The training procedure consists of a fixed number of steps K per episode. At each step k, the agent can modify the adversarial vehicle's trajectory by adjusting starting point, velocity, and lateral displacement. Subsequently, the motion planner solves the constructed scenario, and the agent receives a new observation and reward. After successful episodic training, from updates of the agent in lines 17 and 18, its policy network $\pi(\boldsymbol{o})$ can be inferred with scenario observations $\boldsymbol{o}$ to adjust adversarial trajectories $\boldsymbol{Z}_{xy}^{adv}$ automatically. Additionally, new scenarios can be sampled from the experience replay buffer $\mathcal{D}$, which contains previously collected scenarios during training.

IV. Experiments

In this section, we present the scenarios generated from our proposed trajectory generation framework. We provide details about the experimental setup, including the selected intersection scenario from the CommonRoad [13] repository and the results obtained.

A. Experimental Setup

1) CommonRoad Scenario: For our experiment, we selected a specific CommonRoad Scenario that focuses on a large intersection with multiple lanes and a single dynamic obstacle. The intersection is designed with a total of five major lanes, offering diverse paths for vehicle movement. Two lanes span from left to right, two from right to left, an additional pair of lanes facilitate left turns in upward and downward directions, respectively, as well as one turning right in downward direction from the left. Each lane has a defined direction of travel, ensuring realistic traffic flow within the scenario. The CommonRoad vehicle model of a BMW 320i to model the ego vehicle's and adversarial vehcile's dynamic behavior.

2) Metric: The criticality of the generated scenarios is evaluated using the WTTC [10]. The WTTC aims to get the worst-case approximation of all situations, thereby neglecting as few critical situations as possible. Based on the selected criticality metric the agent's observation (7) is modified as follows:

$$\boldsymbol{o} = \begin{bmatrix} \boldsymbol{Z}_{xy}^{adv} & \boldsymbol{Z}_{xy}^{ego} & \boldsymbol{WTTC} \end{bmatrix} \tag{18}$$

$$= \begin{bmatrix} x_{adv}^{0\ldots\mathcal{T}} & y_{adv}^{0\ldots\mathcal{T}} & x_{ego}^{0\ldots\mathcal{T}} & y_{ego}^{0\ldots\mathcal{T}} & wttc^{0\ldots\mathcal{T}} \end{bmatrix}, \tag{19}$$

where $x_{adv}, y_{adv}, x_{ego}, y_{ego}$ represent the x and y coordinates of the adversarial and ego vehicle trajectory, and $wttc$ the value of the criticality measure starting from timestep 0 until $\mathcal{T}$. Accordingly, the risk metric r_{metric} (15) of the reward function (14) is also adjusted:

$$r_{\text{metric}} = \min\left(\boldsymbol{WTTC}^2\right). \tag{20}$$

For the following experiments the weightings of the reward function are set to $\{b_1, b_2, b_3\} = \{10, 1, 1\}$.

3) Reinforcement Learning Agent: We implement the proposed algorithm and its architecture, as depicted in Fig. 2, using the deep RL framework *StableBaselines3* [31]. Details regarding the selected hyperparameter are listed in Table I. The environment, including the integration with the CommonRoad scenario simulation, is built upon the *Gymnasium* package [32].

B. Experimental Results and Discussion

We demonstrate the effectiveness of our deep RL approach in generating challenging yet solvable scenarios causing planners almost to collide and drive uncomfortably. The RL agent is trained on 15 routes extracted from the selected intersection scenario. In our experiments, the agent can vary the adversarial vehicle's speed from $5\frac{m}{s}$ to $20\frac{m}{s}$, the initial position between the start and end of the elected route and the lateral displacement $2m$ to the left and right of the lanelet's center. We analyze and present the selectively collected scenarios obtained during the training process.

1) Criticality Evaluation: We evaluated the scenarios collected in the experience replay buffer during training of the RL agent according to the WTTC. Fig. 3 illustrates the qualitative results for three exemplary scenarios and the corresponding WTTCs values generated based on the given training routes. In both scenarios in Fig. 3a and 3b, the adversarial vehicle is approaching the ego vehicle. In both cases, the ego vehicle tries to avoid the adversarial vehicle, resulting in a near collision and a low WTTC. In Fig. 3c, the adversarial vehicle approaches the ego vehicle from the right. Due to the consistently small distance throughout the scenario, this results in a high probability of a collision and a low WTTC. Our results show that the RL agent is able to simultaneously generate adversarial trajectories for different routes that result in scenarios with low WTTCs in places or throughout the scenario.

2) Displacement Evaluation: In this section, we establish a baseline for comparison. The baseline trajectory $\boldsymbol{Z}_{xy}^{ego,ref}$ is obtained by the motion planner in the empty scenario without other actors. We compare this baseline trajectory $\boldsymbol{Z}_{xy}^{ego,ref}$ with the ego vehicle's trajectory $\boldsymbol{Z}_{xy}^{ego}$ in the presence of the

TABLE I
SAC HYPERPARAMETERS

Hyperparameter	Value
optimizer	`Adam`
number of samples per batch	128
learning rate (λ)	3×10^{-4}
gradient steps (g)	4
policy delay	2
replay buffer size	1×10^6
discount (γ)	0.95
entropy target	$-\dim(\boldsymbol{a})$
entropy temperature factor (α)	1
target network smoothing coefficient (ρ)	5×10^{-3}
number of hidden layers	2
number of hidden units per layer	256
number of LSTM layers	1
number of hidden units per LSTM layer	128
activation functions	`ReLU`
dropout rates	0.02

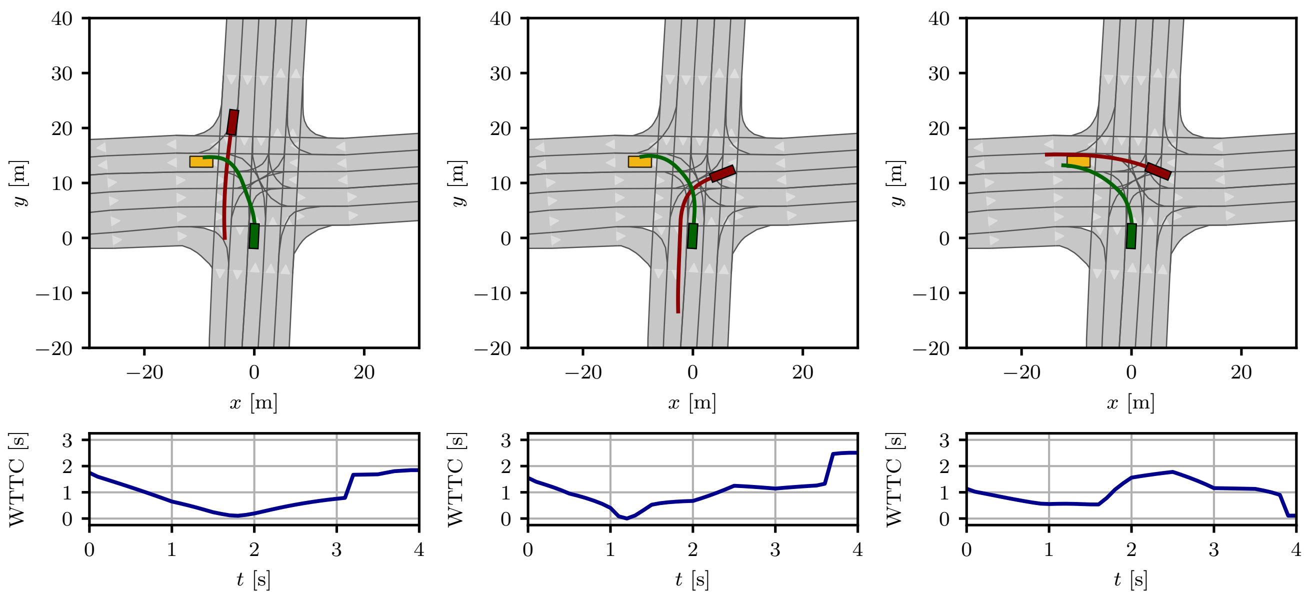

Fig. 3. Example scenarios from the proposed trajectory generation approach evaluated according to the WTTC. It shows the road topology with the lanelets in grey, the contours of the lanelets in black, the ego vehicles planning goal in yellow, and all agents in the scene with their future trajectories. The adversarial agent is shown in red, the ego vehicle is shown in green. A light grey triangle indicates the driving direction in the respective lanelet.

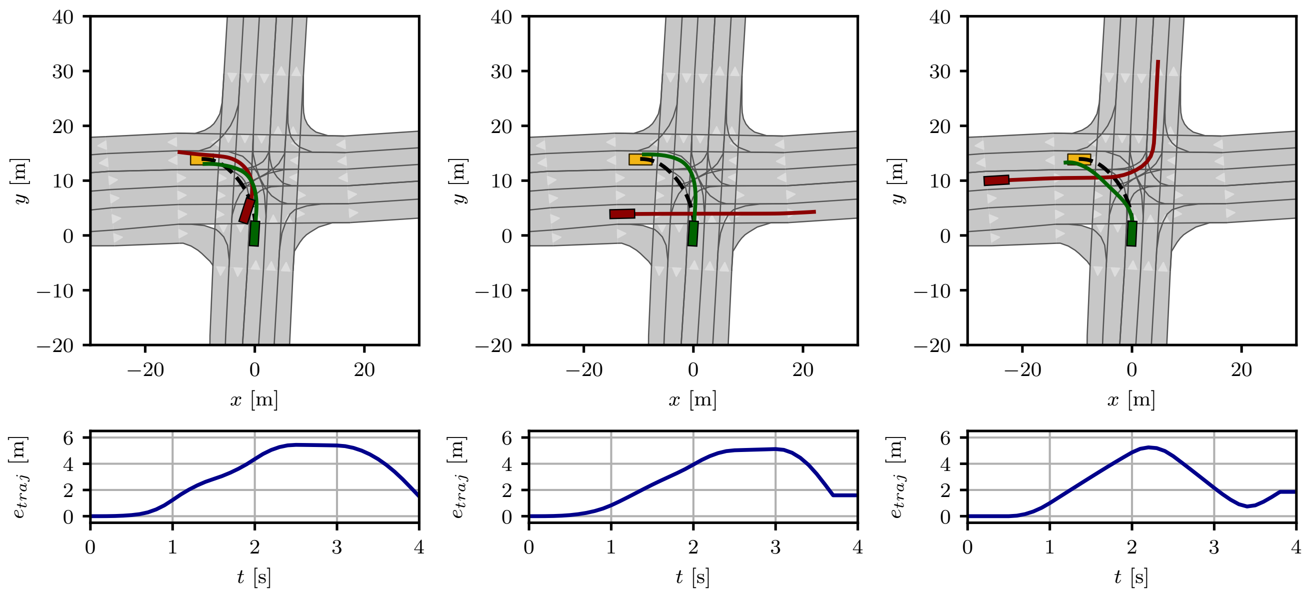

Fig. 4. Example scenarios evaluated according to the error between the ego vehicles trajectory with and without the adversarial agent. It shows the road topology with the lanelets in grey, the contours of the lanelets in black, the ego vehicles planning goal in yellow, and all agents in the scene with their future trajectories. A light grey triangle indicates the driving direction in the respective lanelet. The adversarial agent is shown in red, the ego vehicle is shown in green. The black dashed line indicates the trajectory of the ego vehicles with no adversarial agent existing.

adversarial vehicle. The error e_{traj} between both trajectories in every time step is calculated based on:

$$e_{traj}^{0\ldots\mathcal{T}} = \left\| \boldsymbol{Z}_{xy}^{ego} - \boldsymbol{Z}_{xy}^{ego,ref} \right\|_2 . \tag{21}$$

The experimental results are shown in Fig. 4. In Fig. 4a the ego vehicle is blocked by the adversarial vehicle, which results in a forced standstill at the beginning of the simulation. In order to reach the planning goal in time, the ego vehicle accelerates strongly, followed by heavy braking. In Fig. 4b, the opposing vehicle approaches the ego vehicle from the right. The ego vehicle reacts to this with a high take-off acceleration and a larger trajectory arc, which is characterized by higher lateral accelerations. Similarly, in Fig. 4c, the adversarial vehicle approaches from the left. In order to avoid a collision, the ego vehicle moves evasively, which results in a curve-cutting maneuver. The conducted experiments qualitatively illustrate our approach's ability to generate scenarios, which are characterised by uncomfortable driving behavior, e.g., strong acceleration, heavy braking, and steering. All evasive maneuvers result in high errors e_{traj} compared to the baseline trajectory.

V. Conclusion and Future Work

In this work, we present a novel deep RL approach that aims to explore the space of adversarial trajectories to challenge motion planning algorithms. We demonstrate the effectiveness of our approach by evaluating the generated trajectories based on the Worst-Time-To-Collision (WTTC). Additionally, we investigate the impact on the motion planner by comparing the calculated trajectories with and without the presence of our adversarial agent. However, several interesting questions remain to be addressed in future work on self-driving cars, practical deployment, and testing. Firstly, it is crucial to explore the scalability of our approach to handle multiple dynamic obstacles and other types of obstacles, such as pedestrians. Secondly, the investigation into the design of the reward function for the generation of adversarial trajectories has to be increased to enhance our approach. Lastly, we aim to extend our approach to different classes of motion planners, including RL-based solutions.

References

[1] M. A. Schreurs and S. D. Steuwer, "Autonomous driving-political, legal, social, and sustainability dimensions," *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*, pp. 151–173, 2015.

[2] Council of European Union, "Commission Implementing Regulation (EU) 2022/1426 of 5 August 2022," https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32022R1426#d1e41-20-1, 2022, [Online; accessed 12-May-2023].

[3] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[4] B. Schütt, J. Ransiek, T. Braun, and E. Sax, "1001 ways of scenario generation for testing of self-driving cars: A survey," in *Intelligent Vehicle Symposium, 2023. IEEE.* IEEE, 2023.

[5] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2019, pp. 2352–2358.

[6] B. Schütt, M. Heinrich, S. Marahrens, J. M. Zöllner, and E. Sax, "An application of scenario exploration to find new scenarios for the development and testing of automated driving systems in urban scenarios," *arXiv preprint arXiv:2205.08202*, 2022.

[7] Q. Goss, Y. AlRashidi, and M. İ. Akbaş, "Generation of modular and measurable validation scenarios for autonomous vehicles using accident data," in *2021 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2021, pp. 251–257.

[8] I. Paranjape, A. Jawad, Y. Xu, A. Song, and J. Whitehead, "A modular architecture for procedural generation of towns, intersections and scenarios for testing autonomous vehicles," in *2020 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2020, pp. 162–168.

[9] J. C. Hayward, "Near miss determination through use of a scale of danger," in *Unknown*, 1972, publisher: Pennsylvania State University University Park.

[10] W. Wachenfeld, P. Junietz, R. Wenzel, and H. Winner, "The worst-time-to-collision metric for situation identification," in *2016 IEEE Intelligent Vehicles Symposium (IV).* Gotenburg, Sweden: IEEE, Jun. 2016, pp. 729–734. [Online]. Available: http://ieeexplore.ieee.org/document/7535468/

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning.* PMLR, 2017, pp. 1–16.

[12] P. Cai, Y. Lee, Y. Luo, and D. Hsu, "Summit: A simulator for urban driving in massive mixed traffic," in *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2020, pp. 4023–4029.

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

[14] E. Vinitsky, N. Lichtlé, X. Yang, B. Amos, and J. Foerster, "Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world," *arXiv preprint arXiv:2206.09889*, 2022.

[15] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[17] L. Smith, I. Kostrikov, and S. Levine, "A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," *arXiv preprint arXiv:2208.07860*, 2022.

[18] T. Rudolf, J. Ransiek, S. Schwab, and S. Hohmann, "Robust parameter estimation and tracking through lyapunov-based actor-critic reinforcement learning," in *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society.* IEEE, 2022, pp. 1–6.

[19] D. Baumann, R. Pfeffer, and E. Sax, "Automatic generation of critical test cases for the development of highly automated driving functions," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring).* IEEE, 2021, pp. 1–5.

[20] W. Ding, B. Chen, M. Xu, and D. Zhao, "Learning to collide: An adaptive safety-critical scenarios generating method," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2020, pp. 2243–2250.

[21] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, "Adaptive stress testing for autonomous vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2018, pp. 1–7.

[22] S. Kuutti, S. Fallah, and R. Bowden, "Training adversarial agents to exploit weaknesses in deep control policies," in *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2020, pp. 108–114.

[23] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *2014 IEEE Intelligent Vehicles Symposium Proceedings.* IEEE, 2014, pp. 420–425.

[24] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[25] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[26] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy.* Carnegie Mellon University, 2010.

[27] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka, "Dropout q-functions for doubly efficient reinforcement learning," *arXiv preprint arXiv:2110.02034*, 2021.

[28] J. Stoker, "Differential geometry wiley," 1969.

[29] G. E. Farin, *NURB curves and surfaces: from projective geometry to practical use.* AK Peters, Ltd., 1995.

[30] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "Abcnet: Real-time scene text spotting with adaptive bezier-curve network," in *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9809–9818.

[31] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[32] Farama Foundation, "Gymnasium," https://gymnasium.farama.org/, [Online; accessed 26-May-2023].