Transportation Research Part C 139 (2022) 103662

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

# Autonomous navigation at unsignalized intersections: A coupled reinforcement learning and model predictive control approach

Rolando Bautista-Montesano [a], Renato Galluzzi [a], Kangrui Ruan [b], Yongjie Fu [b], Xuan Di [b,c,*]

[a] School of Engineering and Sciences, Tecnologico de Monterrey, Calle del Puente 222, Col. Ejidos de Huipulco, Tlalpan, 14380, Mexico City, Mexico
[b] Department of Civil Engineering and Engineering Mechanics, Columbia University, United States of America
[c] Center for Smart Cities, Data Science Institute, Columbia University, United States of America

ARTICLE INFO

*Keywords:*
Urban navigation
Path planning
Reinforcement learning
Model predictive control

ABSTRACT

This paper develops an integrated safety-enhanced reinforcement learning (RL) and model predictive control (MPC) framework for autonomous vehicles (AVs) to navigate unsignalized intersections. Researchers have extensively studied how AVs drive along highways. Nonetheless, how AVs navigate intersections in urban environments remains a challenging task due to the constant presence of moving road users, including turning vehicles, crossing or jaywalking pedestrians, and cyclists. AVs are thus required to learn and adapt to a dynamically evolving urban traffic environment. This paper proposes a design benchmark that allows AVs to sense the real-time traffic environment and perform path planning. The agent dynamically generates curves for feasible paths. The ego vehicle attempts to follow these paths under specific constraints. RL and MPC navigation algorithms run in parallel and are suitably selected to enhance ego vehicle safety. The ego AV is modeled with lateral and longitudinal dynamics and trained in a T-intersection using the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm under various traffic scenarios. It is then tested on a straight road and a single or multi-lane intersections. All these experiments achieve desirable outcomes in terms of crash avoidance, driving efficiency, comfort, and tracking accuracy. The developed AV navigation system provides a design benchmark for an adaptive AV that can navigate unsignalized intersections.

## 1. Introduction

With expanding tests of Level-4 or Level-5 fully autonomous vehicles (AVs) on public roads (Waymo, 2018; Uber, 2020), future motorway and traffic systems will be composed of both AVs and human road users. While there exists a number of studies on AV control with reinforcement learning (RL) along highways (Wu et al., 2017a,b; Kreidieh et al., 2018), how AVs navigate an urban environment remains a challenging task. Moving road users, including turning vehicles, crossing or jaywalking pedestrians, and cyclists constantly surround AVs in an urban context. Furthermore, navigating and planning in urban environments implies a shorter horizon time than on highways. Relevant navigation parameters, such as pose, acceleration, pose error with respect to waypoints and minimum distance to collision require careful attention in a rapidly changing scenario. This paper aims to develop an integrated redundant planning and control system for a vehicle dynamic model of AVs to safely navigate unsignalized intersections.

* Corresponding author at: Department of Civil Engineering and Engineering Mechanics, Columbia University, United States of America.
*E-mail address:* sharon.di@columbia.edu (X. Di).

https://doi.org/10.1016/j.trc.2022.103662
Received 1 April 2021; Received in revised form 23 March 2022; Accepted 26 March 2022
Available online 21 April 2022

### 1.1. Literature review

Research on AVs has been focused on different branches of artificial intelligence (AI). These approaches aim to develop full environment awareness, more robust perception capability, and full navigation capacities in highly complex environments (Yasuda et al., 2020). In these contexts, model-based approaches would fail due to the level of detail and complexity required to reproduce the behavior of the ego vehicle and entities within the environment. Interested readers could refer to a recent survey paper on the transition from model-based to AI-based control for AVs (Di and Shi, 2021).

Developments in AI are required for online decision making to achieve feasible driving behaviors (Ding et al., 2019). AI techniques attempt to reproduce and simulate the reasoning and learning of a driver. They are capable of dealing with enormous amounts of data with diverse quality and precision. These strategies can be classified into human-like methods, approximate reasoning, logic- and heuristic-based (Claussmann et al., 2017). Human-like methods intend to imitate human decision processes to solve the navigation problem or to generate driver models. They are commonly used for predictive and reactive planning (Salvucci, 2006). Approximate reasoning methods excel in classifying non-Boolean new knowledge and adapting to future situations. They can develop an intelligent behavior through demonstrations and explanations. These methods are based on logic and statistics (Bojarski et al., 2016). Logic-based approaches are expert systems that require a rule knowledge base and an inference system to generate an intelligent Boolean output. High precision discrete models are needed for these techniques to work properly (Li et al., 2017). Heuristic algorithms intend to find a complete or approximate set of actions to the navigation problem. Their low computational cost in semi-complex problems makes them popular. However, they require an expert guideline to find a local solution. This can lead to losses in optimization and accuracy (Li et al., 2018).

RL constitutes one of the three branches of machine learning, the others being supervised and unsupervised learning. An RL agent can learn through interactions with its environment. It perceives its states in the environment and performs an action at every time step to transition into a different state. Thus, the agent intends to cumulatively maximize its reward through interactions (Zhang and Yu, 2020). In the context of autonomous driving, RL has lead to promising results. The work presented by Isele et al. (2018) describes an RL policy that outperforms typical heuristic approaches. Bouton et al. (2019) show a solution based on a discretized environment that employs a learning approach with belief updates. In RL, the problem to resolve is described as a Markov decision process (MDP). Kamrani et al. (2020) propose an MDP to understand when and how throttle maneuvers should be performed. Furthermore, stochastic predictive control has been successfully applied to partially observable Markov decision processes (POMDP) (Li et al., 2019). Multiple-robot systems have also been trained using RL techniques. For example, complex centralized environments provide the opportunity to validate how simulation-trained robots can perform well in physical scenarios (Fan et al., 2020).

Although deep learning (DL) has been recently used for mobile robotics (Gromniak and Stenzel, 2019; Gu et al., 2020), its combination with RL has leaned rapidly towards solving high complexity problems in robotics (Liu et al., 2020) and, specifically, in self-driving cars (Yaqoob et al., 2020). Deep reinforcement learning (DRL) methods are well suited to generalize and scale interactions with a car's environment and other agents (Kalashnikov et al., 2018). Studies have demonstrated and verified DRL performance when generating optimal navigation policies with high-dimensional sensors (Zeng et al., 2020). In the context of autonomous driving, the work presented by Ma et al. (2021) described the interaction among vehicles and other scenario elements in a T-intersection. Their learning framework was enhanced with supervised learning by actively using a knowledge base.

RL is a suitable tool for AV navigation in driving environments with diverse road layouts and number of vehicles on the road. However, a major challenge the RL techniques face nowadays is safety. The RL-based agent should not violate safety constraints. Policies and actions must be designed such that the agent can produce a safety-enhanced behavior that avoids threatening situations. Combining RL with other algorithms has proven to be a reliable solution to strengthen its performance (Dulac-Arnold et al., 2021). Model Predictive control (MPC) is an optimization-based technique that can resolve these issues for nonlinear dynamics and uncertain constrained systems. This control strategy requires an accurate plant model to develop acceptable predictions. Several approaches have been proposed to generate a valid and practical model that can be deployed on physical platforms (Karnchanachari et al., 2020). The works presented in Zanon and Gros (2021), Gros and Zanon (2020), Morinelly and Ydstie (2016) and Zanon et al. (2019) propose variants of the usage of MPC as a function approximator for RL during the exploration and exploitation phases, ultimately guaranteeing safety and stability. The work by Williams et al. (2017) proposes a multi-layer neural network architecture that emulates a dynamic model later incorporated into MPC systems. Further research has been done to achieve asymptotic performance in deep networks like model-free approaches (Chua et al., 2018) and for obstacle avoidance and target tracking (Kordabad et al., 2021). Typical applications of MPC include longitudinal control and path tracking with time restrictions (Chen et al., 2020; Nam et al., 2019) and a combined lateral-yaw-longitudinal coupled control (Lin et al., 2019). Similar approaches use MPC to deal with system state constraints and actuator limitations as a module for an active steering control with direct yaw control strategy (Yang et al., 2019) and as the main nonlinear component to perform evasive maneuvers and avoid rear end collisions (Chowdhri et al., 2021). As a model-based technique, MPC could be a potential candidate to address safety concerns in AI-based navigation strategies like RL (McAllister et al., 2017). Previous efforts have demonstrated that simultaneously running navigation systems can improve the safety performance of an AV (Xu et al., 2018). A main nominal controller provides driving commands during regular and hazard-free situations. The actions of a back-up reactive controller can override the nominal navigation module to ensure an AV safety state (Behere and Torngren, 2015; Kiran et al., 2021). Although diverse control techniques have been merged by following the described concept, there is still an area of opportunity to explore this approach with AI-based navigation modules. In fact, this kind of system architecture is one of the challenges that navigation on AVs faces (Amer et al., 2016).

Most research on AI relies on specific physical and virtual scenarios to benchmark the proposed algorithms (Vinitsky et al., 2018; Claussmann et al., 2020). Continuous action and observation space scenarios present an important challenge due to

the required computational overhead to produce meaningful results. To tackle this shortcoming, recent works rely on model discretization (Marchesini and Farinelli, 2020). Novel discretized RL-based methods like ANOA, NavMesh, Robust Action Governor (RAG), or MaxPain have been proposed to generate safe paths for autonomous systems that can navigate in a wide range of environments (Wu et al., 2020; Alonso et al., 2020; Li et al., 2021; Wang et al., 2021). Furthermore, accurate modeling of the plant introduces an additional behavior that is accounted for during training. This dynamic content may affect the performance of the applied control strategy. Recently, the works presented in Baheri et al. (2020a) and Baheri et al. (2020b) show the potential use of a continuous state space with discrete actions and a sensor to measure the environment.

### 1.2. Modeling framework

The approach presented in this work is illustrated in the block diagram of Fig. 1. It shows an ego vehicle that captures observations $o$ through sensing and yields actions $a_{1,2}$, which are the propulsion force of the vehicle and a local path to follow. The RL scheme formulates a reward $r$ as a function of the vehicle state $s$ and actions $a_{1,2}$. Unlike previous efforts in literature, this work approaches the problem using a more realistic representation of the plant by means of its longitudinal and lateral dynamics, which play a fundamental role in the responsiveness of the ego vehicle. Due to causality in vehicle dynamics, the applied action will produce a variation in the vehicle's pose. To complement the RL action, the lateral control – a pure-pursuit controller (Coulter, 1992) – determines the steering angle to apply to the vehicle based on dynamically generated target waypoints. These waypoints come from a Frenét frame-based path-planning algorithm (Werling et al., 2010). Despite addressing different degrees of freedom (DOFs) and levels of the ego vehicle, the different control actions are intrinsically coupled. In fact, the effectiveness of the pure-pursuit control depends on the so-called look-ahead distance, which is proportional to the vehicle longitudinal speed. On the other hand, the bandwidth of the trajectory planning algorithm is affected by the rate at which the pose of the vehicle changes in time. Moreover, the obtained waypoints and the steering angle will influence the reward in RL, as they will affect the performance of the vehicle while reaching its final destination. A simultaneous MPC module runs a lateral path tracking system and a longitudinal cruise control system. A safety system uses environmental and internal variables to determine if the RL or MPC lateral/longitudinal system controls the ego vehicle.

The proposed model follows a hierarchical approach (Fig. 1). First, the environment is perceived by internal and external sensors: LIDAR and odometer. Sensor measurements are processed by a perception module. These data are transformed into information that describes the size and location of each element within sensor range of measurement. This information is fed to the path planning module. The local planner information is transferred to a control module, which is constituted by RL and MPC modules that run simultaneously. Each module generates command signals for the lateral and longitudinal behavior of the vehicle. Depending on the state of the environment a binary safety submodule selects what navigation algorithm should take control of the ego vehicle. This module determines whether the safety distance with the surrounding actors is kept. Collected information from the ego vehicle and the environment are passed to the continuous reward function to evaluate the performance of the ego vehicle according to the specified constraints. Finally, the control signals are sent to the 3-DOF dynamics module of the vehicle. The actions performed by the vehicle affect the environment. Hence, this set of elements closes the control loop.

### 1.3. Contributions of this paper

The major contributions of this paper are listed as follows:

1. This work implements a simultaneously running nominal RL longitudinal control and path selector, and a reactive MPC longitudinal cruise control and lateral path tracker. Both systems can independently control the vehicle lateral and longitudinal dynamics. The coupled RL/MPC architecture represents a redundant fail-safe approach for navigation. Internal and external states are supervised by a safety system. Both algorithms are run in parallel and the control output is selected depending on a safety-oriented discrete selector. In addition, the RL-trained and MPC-enhanced agent can be transferred to environments with different number of vehicles on the road and street layouts.
2. A general RL continuous reward design for AV driving is established, which is a function of longitudinal speed and acceleration, cross-track error, yaw rate, angle to path, distance to collision, and the cost of the selected path. A continuous reward and set of actions provide the agent with complete control over the navigation commands of a vehicle (e.g. steering angle and longitudinal acceleration over the entire planning horizon). This translates into an improved performance in non-discretized or underrepresented environments.
3. A dynamic trajectory planner generates a set of waypoints that the ego AV must select, depending on terminal lateral, longitudinal, and obstacle states. This planner enables the ego vehicle to deviate from the original path by selecting and evaluating a dynamically feasible action. This module helps the training process as the agent does not have to learn how to perform navigation maneuvers; it only selects the most suitable path for a given scenario. The planner also provides the agent with reliable paths when facing unexpected situations that require a course change.
4. The agent controls a three-degree-of-freedom (3-DOF) single-track ego vehicle equipped with internal and external sensors (Fraden, 2016). In this work, vehicle dynamics is taken into consideration for the lateral and longitudinal DOFs, which contrasts the previous effort that the vehicle is modeled as a point governed by kinematic laws.

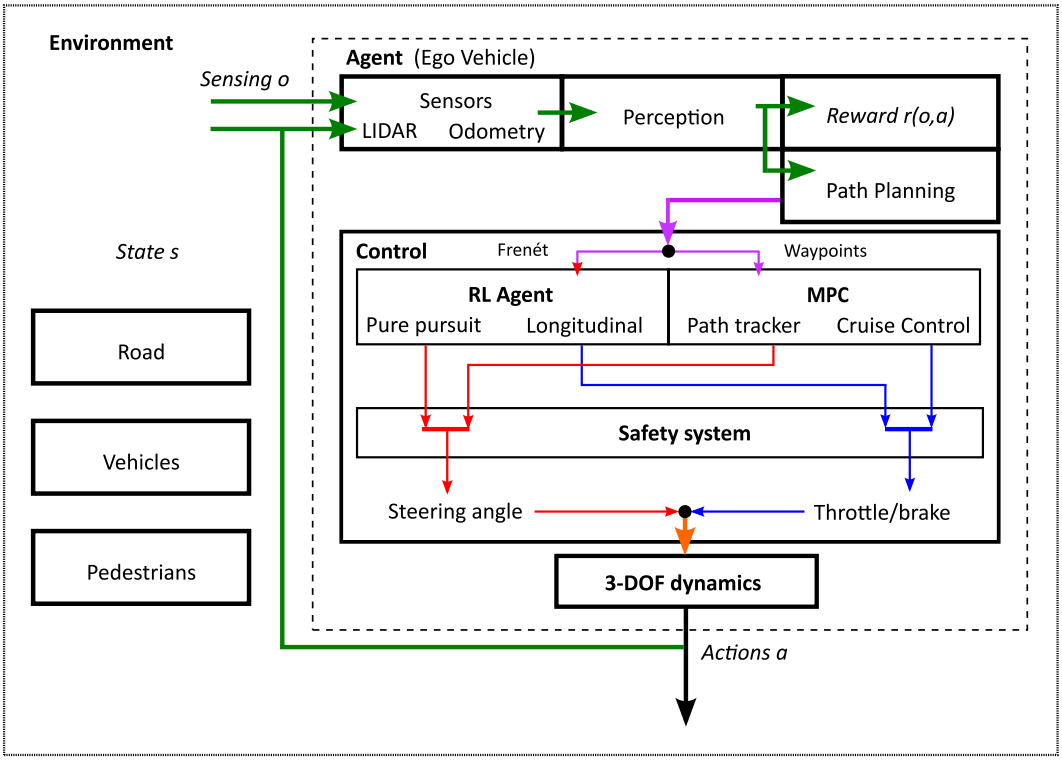



**Fig. 1.** AV navigation block diagram.

In summary, the integrated RL and MPC models, the continuous reward functions, the dynamic waypoint generation, and the dynamic vehicle model, all add more realism to the autonomous vehicle control.

The paper is organized as follows: Section 2 presents the theoretical framework used for the Frenét path planning algorithm, and the 3-DOF vehicle model. the RL TD3 algorithm used for the longitudinal control and path selector agent can be found in Section 4. Section 5 introduces the MPC system used for path tracking and cruise control, and the fail-operational module in charge of selecting the appropriate navigation algorithm depending on the current state. Section 6 explains the rationale for the experiment design. Experiment design includes the definition of the environment: layout and traffic elements, states and observations, sensors, actions, lateral control, and reward design. It also contains the results of the experiment: resulting training curve, navigation demonstration of the RL and MPC systems, and the transferability to other environments. Finally, Section 7 concludes this work and proposes future research directions. For notation clarity, a nomenclature list is provided in Appendix A.

## 2. Path planning

The path planning method for the model is selected after a state-of-the-art comparison among different algorithms. The review presented by Claussmann et al. (2020) highlights three algorithms that are mostly used in path planning for navigation: Hybrid A*, Rapidly-exploring Random Tree (RRT), and Frenét Trajectory. Hybrid A* and RRT are pathfinding algorithms, i.e. they are a subset of graph theory dedicated to solve combinatorial problems through the optimization of a cost function.

Hybrid A* is a variation of A* algorithm for known environments. It applies the Dijkstra algorithm with a heuristic search procedure to expand the fewest possible nodes to the goal in aims of achieving an optimal path. This approach considers kinematic constraints for nonholonomic vehicles. It requires a finely tuned heuristic. The graph should be recomputed at every time step (Dijkstra et al., 1959; Boroujeni et al., 2017; Dolgov et al., 2008).

RRT uses the evolution space to create a kinematic feasible path in unknown environments. This algorithm has been used for replanning in several robotics applications. Some works have demonstrated its feasibility in high-speed environments by enhancing RRT with probabilistic optimal nearest neighbors technique. Its main drawback is the poor connectivity that the randomized graph may have, which leads to low replicability (LaValle et al., 1998; Connell and La, 2017; Hwan Jeon et al., 2013).

Semi-parametric and parametric curves methods are the most popular algorithms used for highway navigation. The main reasons are that roads can be abstracted as a succession of simple and predefined curves, and that a previously defined set of spline curves can contain candidates for a soft constrained path, reactive or predictive maneuvers. Some of these methods directly consider kinematic and dynamic constraints for speed profiles and the continuously changing environment. The curves are a variation of polynomial interpolations that use control points as inflection points. A decision maker function is required to select the most convenient maneuver (Von Hundelshausen et al., 2008; Ziegler and Stiller, 2009; Vanholme et al., 2012; Chen et al., 2013; Lima et al., 2015).

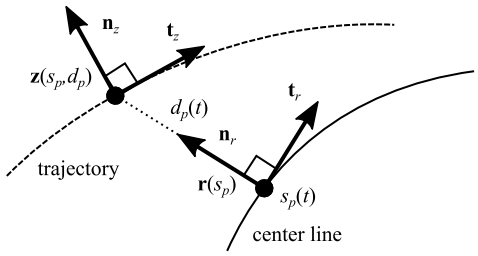



**Fig. 2.** Trajectory generation in a Frenét frame.

### 2.1. Frenét path planner

For proper navigation in realistic environments, the proposed vehicle requires an optimal trajectory generation algorithm (Werling et al., 2010). This module will feed feasible waypoints to the lateral control described in Section 3. In particular, a Frenét frame method is used by following the scheme shown in Fig. 2.

A moving reference frame along the center line is defined by a tangential vector $\mathbf{t}_r$ and a normal vector $\mathbf{n}_r$. This center line is the ideal path to follow. Then, a one-dimensional trajectory is generated for the root point $\mathbf{r}$ and perpendicular offset $d_p$, according to

$$\mathbf{z}(s_p(t), d_p(t)) = \mathbf{r}(s_p(t)) + d_p(t)\mathbf{n}_r(s_p(t)) \tag{1}$$

where $s_p$ is the covered arc length of the center line and $\mathbf{t}_z, \mathbf{n}_z$ are the tangential and normal vectors of the resulting trajectory $\mathbf{z}(s_p(t), d_p(t))$.

Here, we optimize the trajectories by minimizing the cost function $J_t$ given by the time integral of square of jerk $\dddot{p}_p$, the third derivative of position $p_p$ (Takahashi et al., 1989):

$$J_t(p_p(t)) := \int_{t_0}^{t_1} \dddot{p}_p^2(\tau)\, d\tau \tag{2}$$

We assume that the vehicle starts at state $P_0 = \left[p_{p0}, \dot{p}_{p0}, \ddot{p}_{p0}\right]$ at time $t_0$ and end at time $t_1 = t_0 + T$ in state $P_1 = \left[p_{p1}, \dot{p}_{p1}, \ddot{p}_{p1}\right]$. It is worth noting that quintic polynomials are jerk-optimal connections between states $P_0$ and $P_1$. The minimization of the cost function

$$C = k_j J_t + k_t g(T) + k_p h(p_{p1}) \tag{3}$$

with defined terms $g$, $h$ and parameters $k_j, k_t, k_p > 0$ is also a quintic polynomial. We use the cost function form of Eq. (3) to generate trajectories for both lateral sides of the vehicle and the longitudinal DOF.

The described method is summarized as follows. To test a dynamic environment, we determine ground truth of trajectories for each actor to make a prediction. The goal is to generate a feasible and natural optimal path. We generate terminal states that implement three behaviors including cruise control, lane changes, and car following. Among these candidate paths, an optimum path is chosen by considering minimizing cost, kinematic feasibility and collision avoidance. The feasibility of each path is evaluated such that produced lateral dynamics and longitudinal acceleration reference can be properly followed. The generated paths are bound to the vehicle model, which will be described below.

### 2.2. Vehicle model

Vehicles are modeled as a 3-DOF representation known as the bicycle model. Here, the left and right wheels of each axle are lumped into a single wheel. The front axle can steer by means of an angle $\delta$. This angle is considered as an input variable for the plant. Also, the center of gravity of the vehicle is denoted by point C and the distance between the center of gravity and the wheels are denoted with $l_f$ and $l_r$ for front and rear axles, respectively. Accordingly, the wheelbase is defined as $L = l_f + l_r$.

The equations governing the dynamics of this vehicle are

$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_{yf} + F_{yr} \tag{4}$$

$$I_{zz}\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \tag{5}$$

$$m\ddot{x} = m\dot{y}\dot{\psi} + F_{xf} \tag{6}$$

where $m$ is the vehicle mass and $I_{zz}$ is the polar moment of inertia with respect to the $z$ axis. These expressions exhibit a coupled dynamic behavior between the longitudinal ($x$), lateral ($y$) and yaw ($\psi$) DOFs. Furthermore, motion is influenced by the external lateral forces on the front and rear axles ($F_{yf}$ and $F_{yr}$) and the longitudinal force on the front axle ($F_{xf}$). Roll, pitch and vertical dynamics are assumed uncoupled and hence, neglected in this representation.

**Table 1**
Vehicle model parameters.

| Parameter [unit] | Symbol | Value |
|---|---|---|
| Distance from COG to front axle [m] | $l_f$ | 1.4 |
| Distance from COG to rear axle [m] | $l_r$ | 1.6 |
| Mass [kg] | $m$ | 2000 |
| Yaw polar moment of inertia [kg m$^2$] | $I_{zz}$ | 4000 |
| Front tire cornering stiffness [kN/rad] | $C_{\alpha f}$ | 12 |
| Rear tire cornering stiffness [kN/rad] | $C_{\alpha r}$ | 11 |

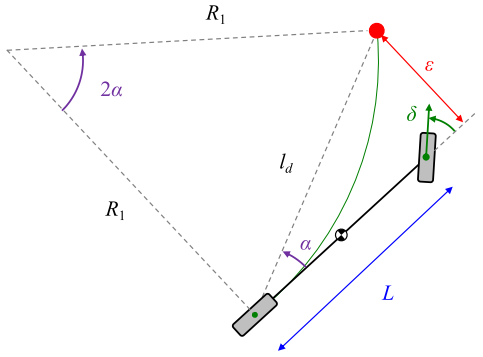



**Fig. 3.** Pure-pursuit kinematic lateral control of the vehicle.

The maximum speed assumed in an urban environment is 20 m/s. Thus, aerodynamic contributions can be discarded for a class-B-sized vehicle. Assuming flat road scenarios and negligible rolling resistance, the longitudinal force component on the front axle $F_{xf}$ contains only a propulsion contribution due to the powertrain of the vehicle. Like the steering angle $\delta$, the propulsion force is an input of the vehicle model.

Lateral force contributions depend on the cornering stiffness of the front and rear tires ($C_{\alpha f}$ and $C_{\alpha r}$):

$$F_{yf} = 2C_{\alpha f}\left(\delta - \theta_{vf}\right) \tag{7}$$

$$F_{yf} = 2C_{\alpha r}\left(-\theta_{vr}\right) \tag{8}$$

where the tire velocity angles are given by

$$\theta_{vf} \approx \frac{\dot{y} + l_f\dot{\psi}}{\dot{x}} \tag{9}$$

$$\theta_{vr} \approx \frac{\dot{y} - l_r\dot{\psi}}{\dot{x}} \tag{10}$$

Lateral dynamics is described by the nonlinear model presented in Eqs. (4) to (6) (Jazar, 2009). Its global expressions can be obtained by substituting Eqs. (9) into (7), and (10) into (8), Finally, resulting Eqs. (7) and (8) should be substituted into (4) and (5) to yield Eqs. (11) and (12).

$$m\ddot{y} = -m\dot{x}\dot{\psi} + 2\left[C_{\alpha f}\left(\delta - \frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}\right) + C_{\alpha r}\frac{l_r\dot{\psi} - \dot{y}}{\dot{x}}\right] \tag{11}$$

$$I_{zz}\ddot{\psi} = 2\left[l_f C_{\alpha f}\left(\delta - \frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}\right) - l_r C_{\alpha r}\frac{l_r\dot{\psi} - \dot{y}}{\dot{x}}\right] \tag{12}$$

The parameters for the vehicle model used in this study are listed in Table 1. This model is used to reproduce the dynamic behavior of the ego vehicle in the RL framework. This representation has been extensively used due to its ability to reproduce accurate vehicle dynamic behavior (Miloradović et al., 2019). In RL, model-based approaches are sample efficient when compared to on-policy gradient methods (Wang et al., 2019). In addition, this model is also used to implement an MPC-based safety module, as demonstrated in Section 5.

## 3. Lateral control

To obtain a steering action, a pure-pursuit kinematic lateral control is proposed based on Coulter (1992). In this strategy, a look-ahead distance $l_d$ is defined between the rear axle of the vehicle and a target waypoint. The goal is to obtain a steering angle $\delta$ that defines a trajectory curvature between the rear axle and the target.

From the representation in Fig. 3, the curvature of the path is determined as

$$\kappa = \frac{1}{R_1} = \frac{2\sin\alpha}{l_d} \tag{13}$$

where $R_1$ is the curved path radius and $\alpha$ is the angle between the vehicle heading and the look-ahead distance vector. Alternatively, this curvature can be seen as a function of the cross-track error $\varepsilon_{ct}$, i.e. the lateral distance between the vehicle heading and the target waypoint:

$$\kappa = \frac{2}{l_d^2}\varepsilon_{ct} \tag{14}$$

From the rear-axle kinematics of the bicycle model, the steering angle can be determined as

$$\delta = \arctan(\kappa L) = \arctan\left(\frac{2L\sin\alpha}{l_d}\right) \tag{15}$$

Hence, the steering angle can be seen as a function of the cross-track error $\varepsilon_{ct}$, which needs to be minimized in order to track the desired trajectory. To ensure a continuous update of the steering angle, the target waypoint needs to move in space as the vehicle navigates through its path. Furthermore, as the vehicle travels faster, this target waypoint must be placed further away from the vehicle to guarantee proper path following. These considerations allow defining a look-ahead distance as a linear function of the vehicle speed $\dot{x}$

$$l_d = K_d\dot{x} \tag{16}$$

and hence, Eq. (15) can be updated accordingly:

$$\delta = \arctan\left(\frac{2L\sin\alpha}{K_d\dot{x}}\right) \tag{17}$$

Thus, the pure-pursuit lateral control defines a steering angle that depends on its longitudinal speed and the waypoints that the vehicle must follow.

## 4. Longitudinal control and path selector

To solve the longitudinal throttle problem and to select an appropriate path, we use the RL framework. Conventionally, there are two kinds of RL methods: value-based methods and policy-based methods. Value-based RL methods, such as Q-learning, exploit the learned value function to guide the selection of the actions. Policy-based methods, e.g. REINFORCE, optimize the cumulative reward directly (Sutton and Barto, 2018). However, value-based RL methods suffer from high dimensional space, and policy-based RL methods are vulnerable to the issue of high variance. To solve those issues, the actor–critic framework was proposed (Konda and Tsitsiklis, 2000; Sutton and Barto, 2018; Shou and Di, 2020; Shou et al., 2022). Subsequently, many RL algorithms, such as Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015), Proximal Policy Optimization (PPO) (Schulman et al., 2017), and Soft Actor–Critic (SAC) (Haarnoja et al., 2018) were built based on this framework.

We propose to use Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm (Fujimoto et al., 2018), a more advanced version of DDPG (Lillicrap et al., 2015). DDPG learns the Q-value function and the policy in the continuous action space simultaneously. Specifically, the Q-value is updated based on Q-learning minibatch updates by minimizing the target and the current Q-values:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,s'}[\underbrace{(r(s,a) + \gamma \max_{a'} Q_{\theta'}(s',a')}_{\text{target}} - Q_\theta(s,a))^2] \tag{18}$$

where $\theta'$ denotes the target network corresponding to the value network $Q_\theta$, $\gamma$ is the discount factor, $\mathbb{E}$ is the expectation, $\mathcal{L}(\theta)$ is the loss function for the Q-value, $r$ is the reward function, $s$ the state, and $a$ the action. With a target network, the algorithm avoids regressing on a rapidly changing value, which makes the learning process much more stable. As for the policy network, following Eq. (19), it is going to learn an approximate maximizer to $a^*(s)$.

$$a^*(s) = \arg\max_a Q^*(s,a) \tag{19}$$

However, DDPG suffers from the issue of overestimation bias (Fujimoto et al., 2018). Because of the gathering numerical error of Q-values, poor states are going to have high values, resulting in unstable and sub-optimal behaviors. Compared to the DDPG algorithm, TD3 is better for fixing function approximation errors both in the Q-value networks and the policy network. There are two Q-value functions in TD3, and the policy is updated less frequently than those two Q-functions. With those mechanisms, TD3 could properly address the issue of overestimation when learning the Q-value functions and have a better performance.

In this research, because the ego vehicle could only obtain the information of other vehicles by sensors, instead of knowing the global information $s$, the actor could only take the observation $o$ as input. And the input to the Q-value network is also based on $o$ and $a$. Then the loss function becomes:

$$\mathcal{L}(\theta) = \mathbb{E}_{o,a,o'}[(r(o,a) + \gamma \max_{a'} Q_{\theta'}(o',a') - Q_\theta(o,a))^2] \tag{20}$$

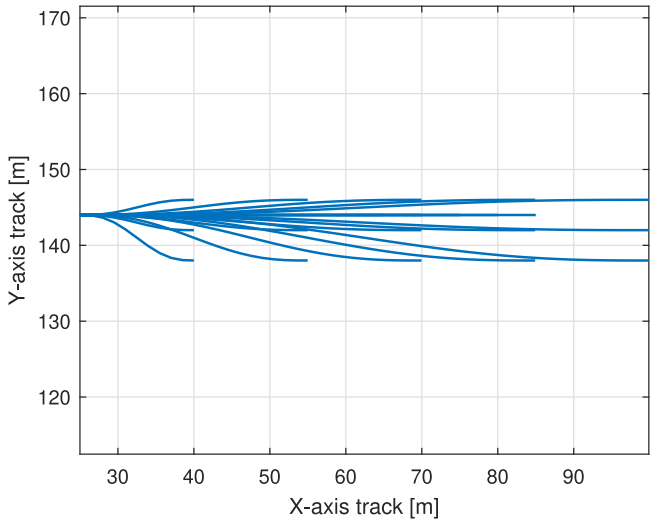



**Fig. 4.** Set of potential paths for $a_2$.

and the approximate maximizer becomes:

$$a^*(o) = \arg\max_a Q^*(o, a) \tag{21}$$

This RL scheme will be applied to solve the navigation at intersections. The state and observation vectors need to be defined according to the environment. The state is denoted as $s = [X, Y, \psi, \dot{x}]$, including non-ego vehicle and environment information. Thus, $s$ is a vector, whose dimension is dependent on the total number of non-ego vehicles. The observation of the proposed POMDP is defined as $o = \left[X, Y, \psi, \dot{x}, \ddot{x}, \varepsilon_{ct}, C_{fp}, \mathcal{Q}_{fp}, \mathcal{P}\right]$. The first two elements ($X \in W_x$ and $Y \in W_y$) are the coordinates [m] of the ego vehicle with respect to the world frame. The third element is the yaw, i.e. the angular rotation of the vehicle with respect to the world-fixed $Z$-axis ($\psi : [-\pi, \pi]$). The next two elements are the longitudinal speed [m/s] and acceleration [m/s$^2$] of the vehicle along its local $x$ axis ($\dot{x}$ and $\ddot{x}$). The cross track error ($\varepsilon_{ct}$) is the distance in meters from the desired path to the position of the vehicle. The cost function of the selected trajectory $C$ evaluates the selected Frenét path in terms of time, longitudinal, and lateral deviation. The collision checker $\mathcal{Q}_{fp}$ is a Boolean operator that verifies whether the selected path leads to a collision state. Finally, $\mathcal{P}$ is a flattened vector that contains the states of the elements surrounding the ego vehicle.

Consequently, the agent interacts directly with the throttle and braking (action 1), and steering (action 2) systems of the ego vehicle. Both actions are continuous. Action 1 ($a_1$) controls the power train of the vehicle. This action is defined in the $[-1, 1]$ interval. Before $a_1$ is fed to the vehicle as the input force $F_k$, it is multiplied by a gain $F_{max}$. This gain is constrained by a rate limiter. The throttle and braking system is defined by Eq. (22).

$$F_k = \begin{cases} F_{k-1} + \frac{F_{max}}{5}, & \dot{F}_k \geq \frac{F_{max}}{5} \\ F_{k-1} + \dot{F}_k, & -F_{max} < \dot{F}_k < \frac{F_{max}}{5} \\ F_{k-1} - F_{max}, & \dot{F}_k \leq -F_{max} \end{cases} \tag{22}$$

Action 2 ($a_2$) selects one of the $N$ potential trajectories generated by the path planner presented in Section 2. Its action space is defined in the $[-1, 1]$ interval. Each value of $a_2$ maps to one of the calculated Frenét paths. The origin of each path $\mathcal{F}_i$ is located at the current ego vehicle position. Each endpoint ranges from the center of the left adjacent to the center of the right adjacent lane (Fig. 4).

*Actor network.* The actor takes the observation $o$ as input, and outputs the optimal action, $a^*(o)$, with respect to the approximate maximum Q-value. Due to the complicated continuous action space and observation space, we decide to use a neural network by utilizing its property of universal function approximator. It is also called the policy network, denoted as $\pi_\phi(o)$. See Table B.6 for more implementation details. Because the action space is continuous, it is intractable to calculate the maximum Q-value according to $a$ directly. Therefore, the actor uses the deterministic policy gradient to update its parameters:

$$\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_a Q_{\theta_1}(o, a) \nabla_\phi \pi_\phi(o) \tag{23}$$

where $Q_{\theta_1}(o, a)$ is one of the Q-value functions, and $\nabla_\phi \pi_\phi(o)$ is the output gradient of the policy network with respect to itself.

*Critic network.* There are two critic networks in the TD3 algorithm (see Table B.7). They take the observation $o$ and the action $a$ as input, and output the Q-value for this observation-action pair. When calculating the value of the targets, TD3 uses the smaller value of those two Q-values, which helps addressing the issue of Q-value over-estimation:

$$y \leftarrow r + \gamma min_{i=1,2} Q_{\theta_i'}(o', \tilde{a}) \tag{24}$$

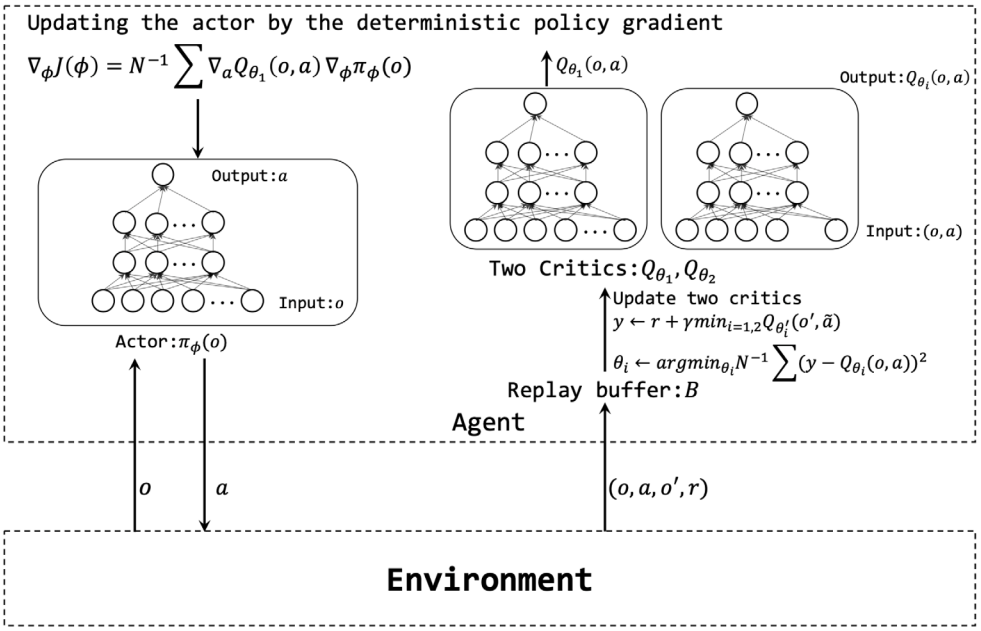



**Fig. 5.** A simplified version of TD3 only including the actor and two critic networks.

where $o'$ is a mini-batch of next observations sampled from the replay buffer, $Q_{\theta_i'}$ is the target network, and $\tilde{a}$ is generated by the target policy network. Denoting these two Neural Networks as $Q_{\theta_1}(o,a)$ and $Q_{\theta_2}(o,a)$ allows for approximation of current model $Q(o,a)$. Those two critic networks are updated by minimizing the loss below:

$$\frac{1}{N} \sum (y - Q_{\theta_i}(o,a))^2 \tag{25}$$

To summarize, Fig. 5 presents a simplified version of the algorithm to clarify the relationships among the environment, the actor, and two critic networks. The TD3 algorithm for longitudinal control is summarized in Algorithm 1 (adapted from Fujimoto et al. (2018)). It begins with the initialization of the replay buffer, two critic networks, and a policy network. With each neural network, a corresponding target network is initialized, following the logic of line 2 in Algorithm 1. At each time step $t$, an action $a_t$ is selected from $\pi_\phi$ with exploration noise $\epsilon$. By executing $a_t$, the reward $r_t$ and the new observation $o_{t+1}$ are observed, composed of a tuple $(o_t, a_t, r_t, o_{t+1})$, which is later stored in the replay buffer $\mathcal{B}_R$. With random sampling of a mini-batch of transitions from the replay buffer, two critic networks are updated by minimizing the matching loss. At a slower rate $T_{update}$, the policy network is updated based on the sampled deterministic policy gradient, and the target networks are updated with respect to the parameters of the corresponding networks.

The vehicle dynamics model, the pure pursuit controller, and the RL training algorithm for longitudinal control and path selector are combined to generate a TD3-based agent that can navigate in different road layouts and number of vehicles on the road. They provide the agent with a realistic dynamic model, feasible paths to follow, and navigation constraints. This helps the RL algorithm maintain a more stable performance. This framework can have drawbacks when facing unexpected situations. A fail-operational system based on MPC, which will be described in Section 5, is proposed to handle these events and ensure the safe navigation of the ego vehicle.

## 5. Safety enhancement

To enhance the safe operation of the ego vehicle, a MPC strategy is run in parallel to the RL-based method. The proposed nominal RL-based navigation system has a broad field of action. It can select any of the calculated Frenét paths and can modify the vehicle's acceleration profile according to the selected maneuver. Whenever a critical situation is identified, the safety enhancement system will switch from RL to MPC until the detected hazard is avoided. MPC will provide hard-constrained navigation and maneuverability features: safe lane keeping and cruise control. This module is explained in Section 5.1. Its simplicity aims to reduce the computational load of the overall system, as well as to increment the ease of implementation.

### 5.1. Model Predictive Control (MPC)

MPC is based on real-time model execution and optimization to yield the most suitable command to the controlled plant. To this end, expressions (11) are (12) are linearized and integrated with a first-order longitudinal model:

$$\ddot{x} = -\frac{1}{\tau_a}\ddot{x} + \frac{1}{\tau_a} a_{ref} \tag{26}$$

**Algorithm 1** TD3: Twin Delayed Deep Deterministic policy gradient algorithm for longitudinal control

Randomly initialize two critic networks $Q(o,a|\theta_1), Q(o,a|\theta_2)$ and policy network $\pi(o|\phi)$ with weights $\theta_1, \theta_2, \phi$
Initialize target networks $Q'(o,a|\theta'_1), Q'(o,a|\theta'_2), \pi'(o|\phi')$ with weights $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialize the Replay Buffer $\mathcal{B}_R$
**for** epoch $= 0, 1, 2, ..., M$ **do**
    the initial observation: $o_0 = (X_0, Y_0, \psi_0, \dot{x}_0, \mathcal{P}_0)$ defined in Section 4
    **for** t $= 0, 1, 2, ..., T$ **do**
        Select action $a_t = \pi_\phi(o_t) + \epsilon$, i.e. the longitudinal force $F_{xf}$, according to $\pi_\phi$ with exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$
        Execute $a_t$, and observe the reward $r_t$ and obtain the new observation $o_{t+1} = (X_{t+1}, Y_{t+1}, \psi_{t+1}, \dot{x}_{t+1}, \mathcal{P}_{t+1})$ from the new state $s_{t+1}$
        Store transition $(o_t, a_t, r_t, o_{t+1})$ into $\mathcal{B}_R$

        Sample a size N mini-batch transitions $(o, a, r, o')$ from $\mathcal{B}_R$
        Set $\tilde{a} \leftarrow \pi_{\phi'}(o') + \epsilon, \epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$ by the target policy network
        Set $y \leftarrow r + \gamma min_{i=1,2} Q_{\theta'_i}(o', \tilde{a})$ by the target critic networks
        Update each critic $Q_{\theta_i}, i = \{1, 2\}$ by minimizing the loss $\frac{1}{N}\sum\left(y - Q_{\theta_i}(o,a)\right)^2$
        **if** $t \mod T_{update} = 0$ **then**
            Update the policy network $\pi_\phi$ by the sampled deterministic policy gradient:

$$\nabla_\phi J = \frac{1}{N}\sum \nabla_a Q_{\theta_1}(o,a)|_{a=\pi_\phi(o)} \nabla_\phi \pi_\phi(o)$$

            Update the target networks:

$$\begin{aligned}\theta'_1 &\leftarrow \tau\theta_1 + (1-\tau)\theta'_1\\ \theta'_2 &\leftarrow \tau\theta_2 + (1-\tau)\theta'_2\\ \phi' &\leftarrow \tau\phi + (1-\tau)\phi'\end{aligned}$$

        **end if**
    **end for**
**end for**

where $a_{ref}$ is the longitudinal acceleration set-point, and $\tau_a$ is a time lag constant (Rajamani, 2006).

*Model linearization.* A linear time-varying representation is desirable to obtain a simple, yet precise control law able to adapt to different operating conditions. To this end, Taylor series are applied around a linearization point ($\dot{x}_0 = f(x_0, u_0)$).

$$\dot{\tilde{x}} = A(t)\tilde{x} + B(t)\tilde{u} \tag{27}$$

where $A(t)$ and $B(t)$ represent the state and input matrix Jacobians of the nonlinear model. Note that the resulting state-space representation is time-varying, and its variables are expressed around the linearization point $\tilde{x} = x - x_0$; $\tilde{u} = u - u_0$.

The Euler method is used to discretize the model. The discrete state-space representation uses $k$ to abbreviate the time sample:

$$\tilde{x}_{k+1} = A_k\tilde{x}_k + B_k\tilde{u}_k \tag{28}$$

*State prediction.* Let $\tilde{u}_k = \tilde{u}_{k-1} + \Delta\tilde{u}_k$, therefore, (28) can be expressed as:

$$\tilde{x}_{k+1} = A\tilde{x}_k + B(\tilde{u}_{k-1} + \Delta\tilde{u}_k) \tag{29}$$

For convenience, an extended state vector is defined as

$$\xi_k = \begin{bmatrix}\tilde{x}_k\\ \tilde{u}_{k-1}\end{bmatrix} \tag{30}$$

and an output vector $\eta_k$, so that

$$\xi_{k+1} = \tilde{A}\xi_k + \tilde{B}\Delta u_k \tag{31}$$

$$\eta_k = \tilde{C}\xi_k \tag{32}$$

where

$$\tilde{A} = \begin{bmatrix}A & B\\ 0 & I\end{bmatrix},\ \tilde{B} = \begin{bmatrix}B\\ I\end{bmatrix},\ \tilde{C} = \begin{bmatrix}C & 0\end{bmatrix} \tag{33}$$

Note that the obtained representation is written in terms of deviations (Lin et al., 2019). With this approach, an adaptive MPC law is able to yield incremental commands with respect to the previous command. Moreover, linearization points $x_0, u_0$ should be updated at each time step to be consistent with the updated plant model.

Assuming $N_p$ as the prediction horizon and $N_c$ as the control horizon, the compact form of the output is:

$$\eta_k = \Gamma \xi_k + \Theta \Delta U_k \tag{34}$$

where,

$$\eta_k = \begin{bmatrix} \eta_{k+1} & \cdots & \eta_{k+N_p} \end{bmatrix}^\intercal \tag{35}$$

$$\Gamma = \begin{bmatrix} \tilde{C}\tilde{A} & \tilde{C}\tilde{A}^2 & \cdots & \tilde{C}\tilde{A}^{N_c} & \cdots & \tilde{C}\tilde{A}^{N_p} \end{bmatrix}^\intercal \tag{36}$$

$$\Delta U_k = \begin{bmatrix} \Delta\tilde{u}_k & \Delta\tilde{u}_{k+1} & \cdots & \Delta\tilde{u}_{k+Nc} \end{bmatrix}^\intercal \tag{37}$$

$$\Theta = \begin{bmatrix} \tilde{C}\tilde{B} & 0 & \cdots & 0 \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}\tilde{A}^{N_c-1}\tilde{B} & \tilde{C}\tilde{A}^{N_c-2}\tilde{B} & \cdots & \tilde{C}\tilde{B} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}\tilde{A}^{N_p-1}\tilde{B} & \tilde{C}\tilde{A}^{N_p-2}\tilde{B} & \cdots & \tilde{C}\tilde{A}^{N_p-N_c}\tilde{B} \end{bmatrix} \tag{38}$$

Quadratic programming is used to optimize using the following cost function:

$$J_s = \frac{1}{2} e_{k+N_p}^\intercal S e_{k+N_p} + \frac{1}{2} \sum_{i=0}^{N_p-1} \left[ e_{k+i}^\intercal Q e_{k+i} + \Delta\tilde{u}_{k+i}^\intercal R \Delta\tilde{u}_{k+i} \right] \tag{39}$$

Through (39), a constrained minimization problem is established and solved in real time according to the prediction horizon $N_p$ to obtain an optimal command at each time step. A combined MPC framework is used to control the lateral and longitudinal modules of a 3DOF vehicle.

*Adaptive cruise control.* This longitudinal system makes the AV travel at a desired speed as long as a reference distance threshold detector is not triggered. If the detector is activated, the AV keeps a safe distance between the vehicles in its lane. This is, the control tracking target, either the desired $\dot{x}$ or a non-collision distance, is achieved by modifying the throttle of the ego vehicle. The feedback data when the control goal is set to the $\dot{x}$ comes from the odometer, while for the non-collision distance is a function of the processed LIDAR data and the odometer.

*Lane keeping.* The lateral system employs processed LIDAR data that provides the size and location of the lanes to ensure a safe travel withing a road. This MPC system uses road ($\kappa$) and vehicle properties ($\epsilon_{ct}$, $\dot{x}$, relative $\psi$) as feedback data. The controller computes an angle ($\delta$) to minimize $\psi$ respect to the lane centerline and $\epsilon_{ct}$.

### 5.2. Safety module

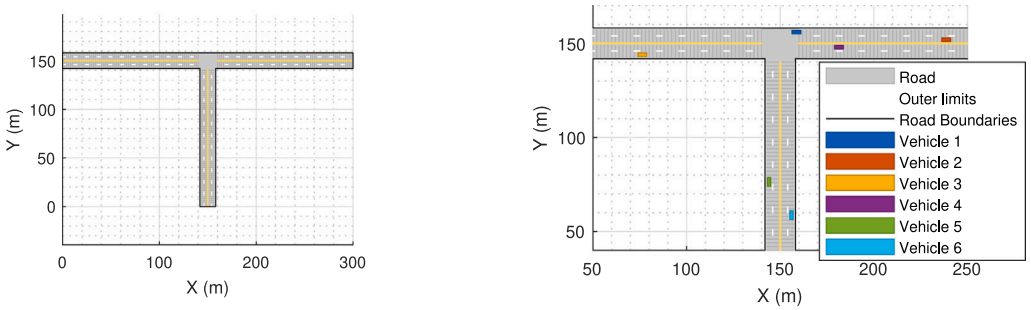
A hard constraint safety node is dedicated to continuously search for critical parameters of the ego vehicle state and its surroundings. Lateral and longitudinal nodes can be triggered separately according to the situation in which the agent is. The safety node switches bidirectionally from RL to MPC lateral and longitudinal commands. A solid-state zero-switching relay mode is used to implement the transitions between states. This type of relay can change between two states with a hysteretic (bistable) nature of two thresholds ($x_L$ and $x_H$) (Fig. 7). State 1 (red) will run and will not change to state 2 (blue) unless $x$ reaches $x_L$. Similarly, state 2 will only change when parameter $x$ reaches $x_H$. Interval $[x_L, x_H]$ is dedicated to ensure less-frequent transitions between states. The changes of state in the switching relay function with hysteresis will determine which algorithm controls the lateral and/or longitudinal elements of the vehicle. This switching function prevents the system to continuously oscillate between navigation algorithms (chattering). This ensures a smoother navigation as each algorithm has sufficient room to control the ego vehicle.

*Longitudinal safety function.* This safety function is defined such that RL is state 1 and MPC is state 2. The parameter that determines the value of the relay triggers is the distance to collision in meters to the leading and trailing vehicles. Triggers are defined as $x_L = 45$ m and $x_H = 50$ m.

*Lateral safety function.* This safety function is defined such that MPC is state 1 and RL is state 2. The function is composed by two parameters: yaw rate $\dot{\psi}$ and cross-track error $\varepsilon_{ct}$. Triggers for the yaw rate are defined as $\dot{\psi}_L = 0.02$ rad/s and $\dot{\psi}_H = 0.04$ rad/s. Similarly, the triggers for the cross-track error are $\varepsilon_L = 3.5$ m and $\varepsilon_H = 4$ m.

## 6. Experiments

This section describes the employed approach to develop a RL agent capable of controlling the throttle and navigation path of a three-degree-of-freedom (3-DOF) vehicle.




(a) Layout (b) Populated with vehicles

**Fig. 6.** T-intersection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

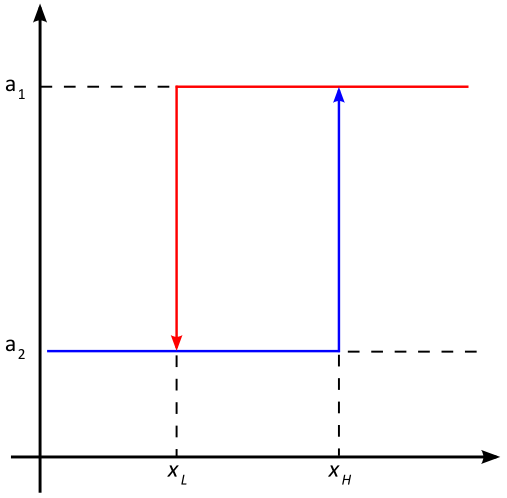



**Fig. 7.** Switching relay function with hysteresis. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### 6.1. Experiment design

We assume that the ego AV can only observe its surroundings using LIDAR sensors, without any connectivity with other objects. The developed work is thus a partially observable Markov decision process. The agent needs to estimate its posterior distributions in potential valid environment states using noisy or incomplete measurements, control effects and environment dynamics (Thrun et al., 2006). This means it is composed of a state $s$ which is not fully observable, observation $o$, reward function $r(o)$, actions $a$, a transition model $p(s'|s,a)$.

This implies that a POMDP on a physical state space can be solved as a traditional MDP on observations. Better decision-making and optimal behavior of a POMDP rely heavily on gathering information from the environment and the agent to reduce uncertainty (Russell and Norvig, 2009). The proposed agent controls an ego vehicle, the properties of which are described in the following subsections.

#### 6.1.1. Environment

The agent controlling the ego vehicle requires a proper environment with diverse potential trajectories. These paths should be as general as possible, such that the agent can be extensible to other scenarios with different layouts and traffic concentration. This section presents the scenario used for training and initial validation of the performance of the agent.

*Street layout and environment characteristics.* The right-hand traffic T-intersection environment is composed of a main road and a side road. These roads have a topology defined as four-lane, two-direction, and no-passing. The main road is 300-m long and the secondary road is 158-m long. Each lane is 4-m wide. The outer parts of the road are delimited by solid white lines. Lanes are divided by double solid yellow lines. Both roads intersect perpendicularly at the middle of the main road (Fig. 6(a)). The intersection area has no lane markings.

The environment and the actors are set to an initial state at the end of each episode. The reset function adds randomness to this setup to avoid overfitting and to generate new valid environments for the TD3-agent.

**Table 2**
Parameters of the traffic elements.

| Parameter [unit] | Vehicle | Bicycle | Pedestrian |
|---|---|---|---|
| Length [m] | 4.7 | 2.2 | 0.24 |
| Width [m] | 1.8 | 0.6 | 0.45 |
| Height [m] | 1.4 | 1.5 | 1.7 |
| Minimum speed [m/s] | 4.16 | 3 | 1 |
| Maximum speed [m/s] | 20 | 4.5 | 1.4 |

**Table 3**
Valid vehicle initial poses.

| Spawn area | $X$ [m] | $Y$ [m] | $\psi$ [deg] |
|---|---|---|---|
| $Z_A$ | $[160, 300]$ | $[150, 158]$ | $\lvert\psi\rvert \geq 150$ |
| $Z_B$ | $[0, 140]$ | $[142, 150]$ | $\lvert\psi\rvert \leq 30$ |
| $Z_C$ | $[150, 158]$ | $[0, 140]$ | $\lvert\psi - 90\rvert \leq 30$ |

*Spawning of traffic elements.* The environment can be populated with three types of elements: pedestrians, cyclists and vehicles. Table 2 shows the main properties of each traffic element. The speed of pedestrians and cyclists was defined according to what was presented in Fitzpatrick et al. (2006). Pedestrians and cyclists usually cross at crosswalks. These areas are bounded by the following coordinates: *Crosswalk 1*: $X \in [158, 160]$ m and $Y \in [142, 158]$ m, *Crosswalk 2*: $X \in [140, 142]$ m and $Y \in [142, 158]$ m, and *Crosswalk 3*: $X \in [142, 158]$ m and $Y \in [140, 142]$ m. Pedestrians can be spawned in the vicinity of any end of a crosswalk and can randomly choose which one to use. Jaywalkers are also considered. These types of element can appear at any side of the street and cross.

Vehicles, cyclists, and pedestrians must have valid poses at the beginning of any episode. Elements are spawned depending on their type. Vehicles should appear on roads with the appropriate heading direction. Table 3 shows ranges of valid poses. Each starting area, as defined by these valid poses, will contain at least two vehicles. The distance between them should be greater or equal to 15 m as long as they are inside the environment boundaries (Fig. 6(b)).

*Vehicle paths and speed profiles.* Vehicles can navigate the environment through valid paths according to their initial spawn area. Given initial spawn areas $Z_A$, $Z_B$, and $Z_C$, let $Z'_A$, $Z'_B$, and $Z'_C$ be the adjacent lanes with opposite direction. Vehicles starting in area $Z_A$ can navigate to $Z'_B$ or $Z'_C$. Similarly vehicles that appear on area $Z_B$ can head to $Z'_A$ or $Z'_C$, and those that with initial pose in $Z_C$ to $Z'_A$ and $Z'_B$. Maneuvers as driving backwards, U-turns, and invading sidewalks or opposite direction lanes are prohibited. Vehicles start navigating at a speed $v \in [v_{min}, v_{max}]$ (See Table 2). Once they arrive at the intersection, they reduce their speed to a maximum of $v_0/2$. Their speed increases linearly until they drive out of the intersection at a speed close to $3v_0/4$. The vehicle speeds up once it reaches its destination area.

### 6.1.2. Sensors

The ego vehicle is aware of the information from other vehicles and pedestrians only through sensors. Below we will detail how sensing data is collected and what information is extracted from which sensor.

*Odometer.* These sensors measure the revolutions of the wheels of the ego vehicle. Depending on their resolution they can measure fractions or full revolutions (Thrun et al., 2006). The ego vehicle is equipped with highly sensitive devices that provide information on its pose $[X, Y, \psi]$ and its time derivatives over a defined sampling time.

*LIDAR.* A Light Detection and Ranging (LIDAR) is an active type of sensor mainly used to measure distances (Chazette et al., 2016). The ego vehicle is equipped with one of these devices. The sensor is placed on the roof of the vehicle and aims towards the front. Its properties can be found in Table 4. The sensor output generates a $32 \times 2250 \times 3$ point cloud. A point cloud is a set of points in a defined space that represent the geometry of an object (Demarsin et al., 2007).

The point cloud elements are clustered by computing the minimum Euclidean distance between points. Each cluster is labeled and the vector $\mathcal{P}_i$ is generated. The vector $P_i$ contains the location in three dimensions $[x, y, z]$ and the size in each dimension $[w, l, h]$ of the detected element.

### 6.1.3. Reward

The navigation agent requires a reward that considers internal and external sensor data. The proposed function considers odometry readings, lateral control error, and the influence of actors in the environment. The complete function is shown in Eq. (40). It is a linear combination of coefficients and nonlinear sub-functions.

Each sub-function is carefully chosen to represent the expected behavior of the longitudinal speed ($r_{\dot{x}}$), longitudinal acceleration ($r_{\ddot{x}}$), cross-track error ($r_{\varepsilon_{ct}}$), yaw rate of the vehicle ($r_{\dot{\psi}}$), angle to path $r_\beta$, distance to collision ($r_d$), and the selected path cost function ($r_{C_{fp}}$).

$$r = K(w_1 r_{\dot{x}} + w_2 r_{\ddot{x}} + w_3 r_{\varepsilon_{ct}} + w_4 r_{\dot{\psi}} + w_5 r_\beta + w_6 r_d + w_7 r_{C_{fp}}) \tag{40}$$

We will subsequently introduce how to compute each component of the reward.

**Table 4**
LIDAR parameters.

| Parameter [unit] | Default setting |
|---|---|
| Update interval [s] | 0.1 |
| Location $x$ [m] | 1.5 |
| Location $y$ [m] | 0 |
| Height [m] | 1.6 |
| Max range [m] | 120 |
| Range accuracy [m] | 0.002 |
| Azimuth resolution [deg] | 0.16 |
| Elevation resolution [deg] | 1.25 |
| Azimuth limits [deg] | $[-180, 180]$ |
| Elevation limits [deg] | $[-20, 20]$ |

*Traveled distance.* The traveled distance to the goal is the first part of the reward function (Eq. (41)). A goal is defined as the terminal state in which the agent completes its navigation task. This function is constituted by the subtraction of two elements: the total length in meters of the path the vehicle needs to travel to reach the goal ($d_t$) and the distance in meters pending to the goal ($d_g$). The first term is computed at the beginning of the run and used subsequently as a constant. The second term is calculated at every instant $k$. During the first instants, $d_t$ and $d_g$ are almost equal and therefore the reward is low. The reward increases linearly as the vehicle travels towards the goal. The only scenario in which this term would award a decreasing or negative value is if it incrementally drives away from the goal. The reward function presented in Eq. (40) is multiplied by this value to provide the agent with a progress estimator (Matignon et al., 2006).

$$K = d_t - d_g \tag{41}$$

*Longitudinal speed.* The dynamic model of the vehicle that is employed considers speeds and forces on the vehicle's $x$ and $y$ axes. The speeds on the $y$ axes can be ignored due to the simulation conditions defined in the environment. The ego vehicle should navigate at acceptable speeds to guarantee continuous traffic. Valid speeds are defined in the interval $\dot{x} \in [5, 20]$ m/s, such that $\dot{x}_{min}$ is the lower boundary and $\dot{x}_{max}$ the upper boundary. Boundaries are expressed as a Boolean function. This function penalizes non-valid speeds, while valid speeds generate a positive value.

$$r_{\dot{x}} = \begin{cases} 2, & \dot{x}_{min} \leq \dot{x} \leq \dot{x}_{max} \\ -1, & \text{otherwise} \end{cases} \tag{42}$$

*Longitudinal acceleration.* A similar approach is used for the longitudinal acceleration. The standard (Secretary, 2019) indicates that an aggressive braking maneuver has a magnitude of 9.65 m/s$^2$ against the motion direction. As the agent should avoid aggressive maneuvers, the valid acceleration interval is defined as $\ddot{x} \in [-5, 1.5]$ m/s$^2$, such that $\ddot{x}_{min}$ is the lower boundary and $\ddot{x}_{max}$ the upper boundary. A Boolean function is also employed (Eq. (43)) with $\ddot{x}_{min}$ and $\ddot{x}_{max}$ as boundaries. This function penalizes non-valid accelerations, while valid accelerations generate a positive reward. For typical automotive applications, powertrains can attain longitudinal accelerations up to 0.5 g, whereas safe braking must not exceed 1 g of deceleration (Genta, 2009).

$$r_{\ddot{x}} = \begin{cases} 1, & \ddot{x}_{min} \leq \ddot{x} \leq \ddot{x}_{max} \\ -1, & \text{otherwise} \end{cases} \tag{43}$$

*Cross-track error.* The difference between a desired location on a trajectory and the measured location of a vehicle is known as the cross-track error (Borhaug and Pettersen, 2005). Minimizing this error results on effecting path following by the agent. A compound Boolean function is employed to compute the cross-track error reward (Eq. (44)). A reward is given to the agent for navigating at the center of each lane $\Lambda_j$. It is acceptable to have small deviations $\lambda$ from the desired path as disturbances can occur. This function penalizes path deviations.

$$r_{\varepsilon_{ct}} = \begin{cases} 2, & \Lambda_j \pm \lambda \\ -1, & \text{otherwise} \end{cases} \tag{44}$$

*Yaw rate.* A similar approach as used for the longitudinal speed and acceleration is proposed for the yaw rate. As the agent should avoid aggressive maneuvers, the valid yaw rate interval is defined as $\dot{\psi} \in [-8, 8]$ deg/s$^2$, such that $\dot{\psi}_{min}$ is the lower boundary and $\dot{\psi}_{max}$ the upper boundary. A Boolean function is also employed (Eq. (45)) with $\dot{\psi}_{min}$ and $\dot{\psi}_{max}$ as boundaries. This function penalizes non-valid yaw rates, while valid values generate a positive reward.

$$r_{\dot{\psi}} = \begin{cases} 1, & \dot{\psi}_{min} \leq \dot{\psi} \leq \dot{\psi}_{max} \\ -1, & \text{otherwise} \end{cases} \tag{45}$$

*Angle to path.* A similar approach as used for the yaw rate is implemented for the heading error to the desired path. The valid maximum interval is defined as $\beta \in [-5, 5]$ deg. A Boolean function is employed (Eq. (46)) with $\beta_{min}$ and $\beta_{max}$ as boundaries. This function penalizes non-valid angles, whereas valid angles generate a positive reward.

$$r_{\beta} = \begin{cases} 1, & \beta_{min} \leq \beta \leq \beta_{max} \\ -1, & \text{otherwise} \end{cases} \tag{46}$$

*Distance to collision.* A dynamic threshold $\tau_c$ is used to compute a safe distance prior to collision. This threshold is proportional to the longitudinal speed $\dot{x}$ of the ego vehicle. A Boolean function is employed (Eq. (47)). This function penalizes risky distances to collision, while safe distances generate a positive reward.

$$r_d = \begin{cases} 1, & d_c \geq \tau_c \\ -1, & \text{otherwise} \end{cases} \tag{47}$$

*Selected path cost function.* The cost function that was presented in Eq. (3) is used to evaluate the performance of the selected path. The cost function considers lateral, longitudinal and time deviations from the global planner path. Positive costs imply a poorly selected path, and negative costs denote a locally optimal path. Thus, the signal value is multiplied by $-1$ to penalize poor paths and reward favorable paths.

$$r_{C_{fp}} = -C \tag{48}$$

### 6.2. Integrated training algorithm

Algorithm 2 outlines the TD3-training integrated with path planning and lateral control, longitudinal control, and safety enhancement module. The results of the presented experiment setup are presented in Section 6.3.

**Algorithm 2** Integrated planning and control algorithm for the ego vehicle

Initialize all the networks mentioned in Algorithm 1
Define ego vehicle and environment (road and traffic elements)
Define vehicle sensors
**for** epoch $= 0, 1, 2, ..., M$ **do**
  Observe the initial state: $o_0 = (X_0, Y_0, \psi_0, \dot{x}_0, \mathcal{P}_0)$ defined in Section 4
  **for** $t = 0, 1, 2, ..., T$ **do**
    Determine local path-planning $g, h, k_j, k_t, k_p$ and lateral control parameters $K_d$
    Run the safety enhancement module to evaluate environment variables to select the navigation algorithm.
    **if** navigation = Frenét **then**
      Generate path planning terminal states for ego vehicle considering cruise control, lane changes and follower vehicles
      Evaluate path-planning cost function $C_t$
      Check trajectories to ensure acceleration and curvature constraints
      Select collision-free optimal waypoints
      Select next waypoint from path according to $l_d = K_d \dot{x}_t$
      Calculate steering angle $\delta_t$ based on next waypoint and vehicle speed $\dot{x}_t$.
    **else if** navigation = MPC **then**
      Select horizon $t$ for path tracking and cruise control
      Calculate the error between the current state with optimal state.
      Calculate steering angle $\delta_t$ and throttle $\ddot{x}$.
  **end if**
  Select actions $a_t = \pi_\phi(s_t) + \epsilon$ according to $\pi_\phi$ with exploration noise $\epsilon \sim \mathcal{N}(0, \sigma)$
  Execute $a_t$, observe the reward $r_t$ and the new state $o_{t+1} = (X_{t+1}, Y_{t+1}, \psi_{t+1}, \dot{x}_{t+1}, \mathcal{P}_{t+1})$ based on the bicycle model
  Store transition $(o_t, a_t, r_t, o_{t+1})$ into $\mathcal{B}_R$
  Sample a size N mini-batch transitions $(o, a, r, o')$ from $\mathcal{B}_R$
  Update each critic $Q_{\theta_i}, i = \{1, 2\}$
  **if** $t \mod T_{update} = 0$ **then**
    Update the policy network $\pi_\phi$
    Update the target networks $\theta_1', \theta_2', \phi'$
  **end if**
**end for**
**end for**

### 6.3. Training results

The TD3-agent was trained in parallel for twenty thousand iterations (16 h approx. of computational time) on a workstation with a 16-core AMD Ryzen 9™ 3950X processor and 64 GB of RAM. The selected hyperparameters can be found in Table B.8. The amount of LIDAR data introduces significant computational burden during training. Computers with low core count processors struggled to train over 100 iterations without reaching a memory allocation crash. The results of the training can be found in Fig. 8. A variable-step solver was chosen for the model to achieve stability due to the continuous nature of the space state and dynamics of the system. The randomized spawn and behavior of the traffic elements, as well as the scenarios are the cause of the distinctive form of the curve.

**Table 5**
Navigation system performance.

| Method | Velocity [m/s] | Acceleration [m/s$^2$] | Distance to collision [m] | Completion time [s] | Success rate [%] |
|---|---|---|---|---|---|
| RL | min: 10.08<br>max: 23.15<br>avg: 18.73 | min: −0.345<br>max: 3.08<br>avg: 2.14 | min: 4.01<br>max: 50.93<br>avg: 31.53 | 10.71 | 94.1 |
| MPC | min: 1.62<br>max: 15.33<br>avg: 5.23 | min: −5.06<br>max: 4.56<br>avg: −0.31 | min: 34.56<br>max: 74.51<br>avg: 61.62 | 29.88 | 97.3 |
| RL/MPC<br>RL duty: 71.1%<br>MPC duty: 28.9% | min: 1.54<br>max: 16.34<br>avg: 6.10 | min: −6.81<br>max: 5.23<br>avg: −0.33 | min: 32.17<br>max: 56.32<br>avg: 43.61 | 26.04 | 99.8 |

The RL and the coupled RL/MPC systems were tested and validated ten thousand times each, in a variety of road topologies: two- or four-lane, two directions, passing or no-passing, single road or T-intersection or crossroad (Sections 6.3.1 and 6.4). A road layout was selected at random for each run. The main road was always 300-m long and the secondary road ranged from 158- to 300-m long. Similarly, a number of 4-m wide lanes was chosen for the given test. Pedestrians, bicycles, and vehicles were spawned with randomized behaviors (Table 2), positions (Table 3), and paths. One to four pedestrians could appear during each test. Jaywalking was one of their potential behaviors if their spawning region was not any of the defined crosswalks. A maximum of two bicycle riders could appear in the environment. Their behavior was set to follow a path on the center of a given lane. However, they could change a lane randomly or stop during navigation. A minimum of two and a maximum of twelve vehicles were spawned in the roads, depending on the given layout and number of lanes. Their initial positions were zones $Z_A$, $Z_B$, $Z_C$, and $Z_D$ accordingly. No more than three vehicles could appear in each zone. A test is considered successful if the ego vehicle navigates from a given initial position to a desired final position without collisions, and the longitudinal velocity and acceleration are between desired limits.

The complete performance data of each navigation system is presented in Table 5. The RL agent successfully completed 94.1% of the tests. The main cause of this success rate is the agent facing extreme situations as pedestrians or other vehicles behaving differently with respect to training. The ego vehicle has a greedy behavior when controlled by RL. This is reflected in the 10.71 [s] it takes to reach the navigation goal. Also, the RL-controlled vehicle violates the maximum speed constraint imposed by the reward function. Its longitudinal control presents low variability in the acceleration command. Finally, the RL driving policy is aggressive, as it can attain a minimum of 4.01 m to collision. By converse, the MPC module had a higher 97.3% success rate when compared to RL. A conservative driving policy yields a safer behavior. This approach has a negative impact on the completion time: 29.88 [s]. The MPC system never surpasses the velocity nor acceleration limits, as both are hard constraints in MPC. There is a high range of variability in the acceleration, with broader ranging among positive and negative values. This kind of behavior could induce chattering, thus hindering passenger comfort. The coupled RL-MPC system achieved a 99.8% success rate. The combination of both navigation modules enhances navigational safety, as the resulting success rate is higher than that of the single control strategies. In addition, the completion time outperforms the slowest technique (MPC). The MPC module controls how greedy the RL system behaves, while maintaining a larger distance to collision with the surrounding elements (32.17 m at minimum). Each system handles the situations in which the counterpart did not perform as expected.

From Table 5, it is worth noting that velocity and acceleration variables for the combined strategy are significantly closer to the metrics of the MPC approach. Therefore, it could be assumed that the MPC safety-enhanced navigation module dominates during vehicle operation. However, a duty cycle analysis of the ten thousand iterations included in Table 5 revealed that the MPC strategy takes control of the ego vehicle for 28.9% of the time. This means that RL-based navigation dominates with a duty cycle of 71.1%. The combination of both strategies can be detected through the distance to collision. In the RL/MPC approach, its minimum is closer to the one of the MPC-based system, whereas its maximum approaches that of the RL-based system. The average distance to collision in the RL/MPC approaches the arithmetic mean of both systems' averages.

### 6.3.1. Navigation algorithm demonstration

The trained TD3-agent was tested in the training scenario on a wide range of combinations of starting points, speeds, and paths. Three samples from those tests are presented: one for the RL, the other for the MPC, and another for the coupled RL/MPC system controlling the ego vehicle.

To give means of comparison, three cases are simulated with the same environment layout. The ego vehicle starts in zone $Z_A$. Its initial pose is set to $\xi_0 = [X, Y, \psi] = [275\text{ m}, 156\text{ m}, 180\text{ deg}]$, and $\dot{x} = 15$ m/s. The path is set to $Z_A \rightarrow Z'_B$. Two extra vehicles are spawned in front of and behind it with a gap of 35 m each. Another vehicle is placed on the adjacent lane. Six extra vehicles are evenly distributed between zones $Z_B$ and $Z_C$ with initial speeds of 12 m/s. The complexity of this path is low as there is a continuous traffic flow in that lane.

The three approaches are evaluated with the following metrics: pose ($\xi = [x, y]^\intercal$), cross-track error $\epsilon_{ct}$, distance to collision, velocity ($\dot{x}$), and acceleration ($\ddot{x}$). These metrics are presented through four graphs in Figs. 9 (RL), 10 (MPC) and 11 (RL/MPC). The pose plot shows the world-based position in meters for $X$ and $Y$ with the lateral distance between the vehicle heading and the target waypoint. The distance to collision plot depicts the Euclidean distance in meters to the closest vehicle surrounding the ego vehicle. The velocity plot illustrates the velocity behavior of the ego vehicle, whereas the acceleration plot presents its throttle/brake activity. The throttle command is bounded by two dashed lines limited at $[-9.6, 4.905]$ m/s$^2$. The outcomes of each test are described in the following paragraphs.

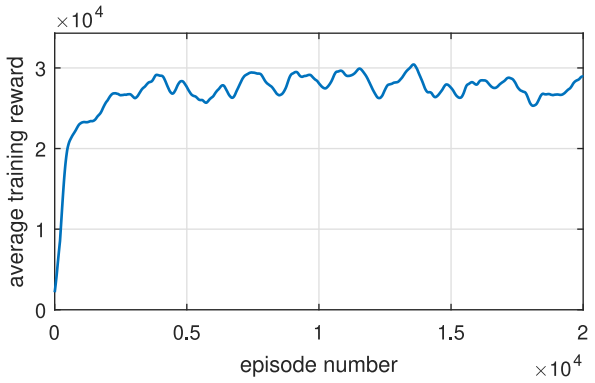
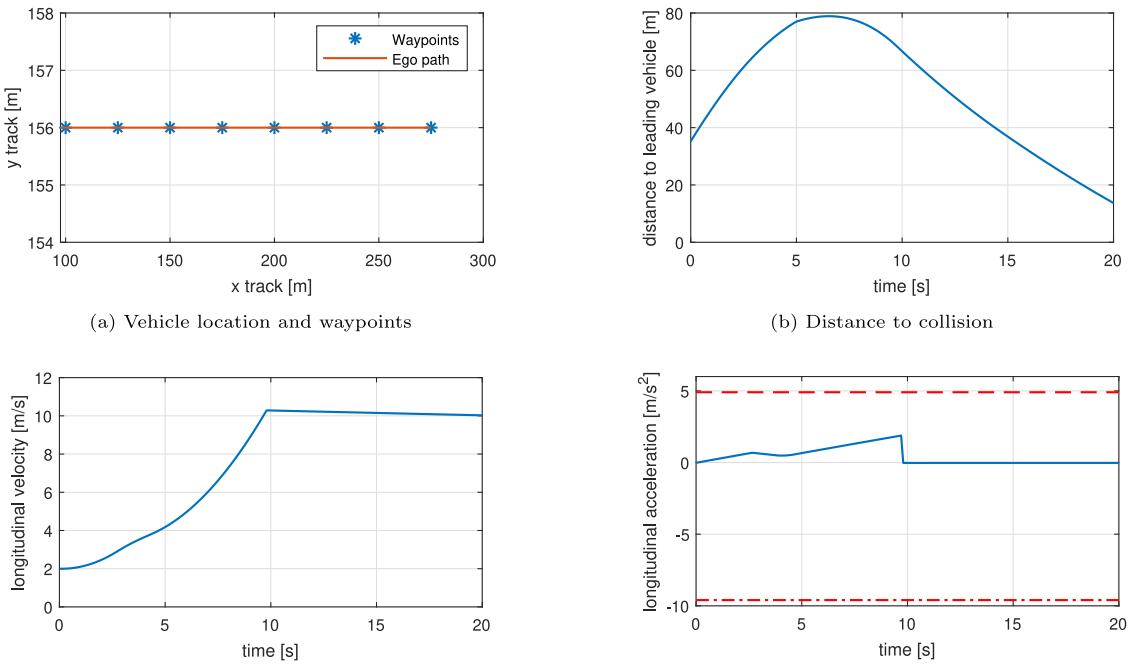



**Fig. 8.** Evolution of the average reward through reinforcement learning training.




(a) Vehicle location and waypoints

(b) Distance to collision

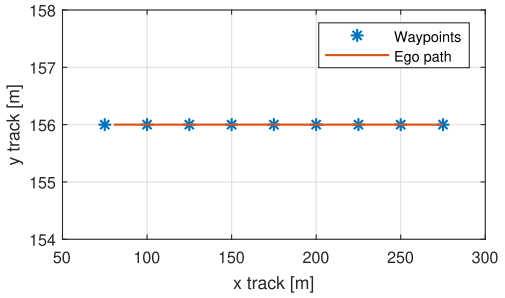
(c) Longitudinal velocity

(d) Longitudinal acceleration

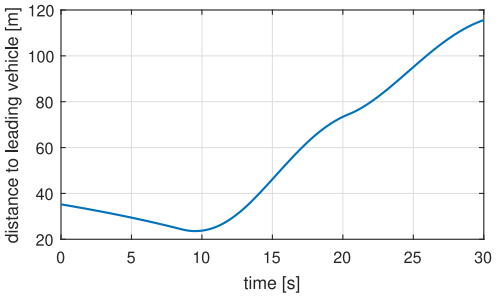
**Fig. 9.** Performance metrics during navigation with RL.

*Test 1 (RL).* It can be noted from Figs. 9(c) and 9(d) how the ego vehicle increases its speed until the distance with the leading vehicle (Fig. 9(b)) reaches a maximum value ($t \approx 7$ s). This encourages the agent to increase the throttle ($t \approx 9.5$ s). The leading vehicle slows down while entering the intersection. Thus, the distance to collision decreases. Causing the acceleration to be set to a value close to zero. This behavior is kept while navigating in the intersection until the goal state is reached. The cross-track error is kept close to zero during the whole navigation maneuver. Also, a minimum safe distance is kept with the leading vehicle and acceleration does not surpass safety limits. Soft constraints are embedded in the RL navigation system through the Frenét path planner.

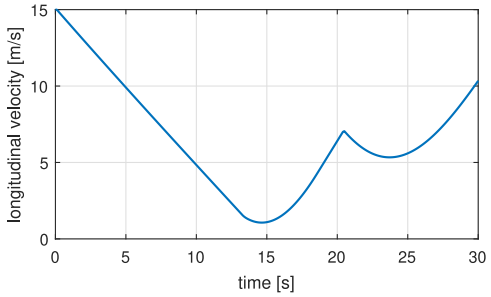
*Test 2 (MPC).* It can be noted from Figs. 10(c) and 10(d) how the ego vehicle decreases its speed until the distance with the leading vehicle (Fig. 10(b)) reaches a minimum value ($t = 10$ s) at the intersection. From this point the speed is slowly increased. The distance to collision with the leading vehicle increases. The cross-track error is close to zero during the whole navigation maneuver. During navigation, safe distance is kept with the leading vehicle. Acceleration does not overcome safety limits. The MPC module implements hard constraints on the navigation system by continuously monitoring the environment. It prevents any risky situation from occurring.



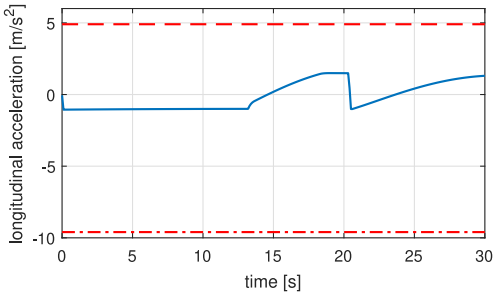

(a) Vehicle location and waypoints




(b) Distance to collision




(c) Longitudinal velocity




(d) Longitudinal acceleration

**Fig. 10.** Performance metrics during navigation with MPC.

*Test 3 (RL/MPC).* Two extra plots are added to check the interaction between the longitudinal (Fig. 11(e)) and lateral (Fig. 11(f)) algorithms. The operation of the safety-enhanced system for longitudinal and lateral commands can be seen through these plots. Section 5.2 explains how the transitions between RL and MPC are done. It can be noted from Figs. 11(c) and 11(d) how the ego vehicle initially decreases its speed to keep a 4 s distance with the leading vehicle (Fig. 11(b)), and to safely navigate the intersection. This behavior is kept until the leading vehicle is far enough. Then speed is increased. Acceleration does not surpass safety limits. The cross-track error is close to zero during the whole navigation maneuver. The coupled RL and MPC system achieved smooth transitions that guarantees full navigation in the test environment. The integrated navigation system implements soft and hard constraints. It prevents the vehicle from going into potential hazardous situations.

### 6.4. Transferability to other environments

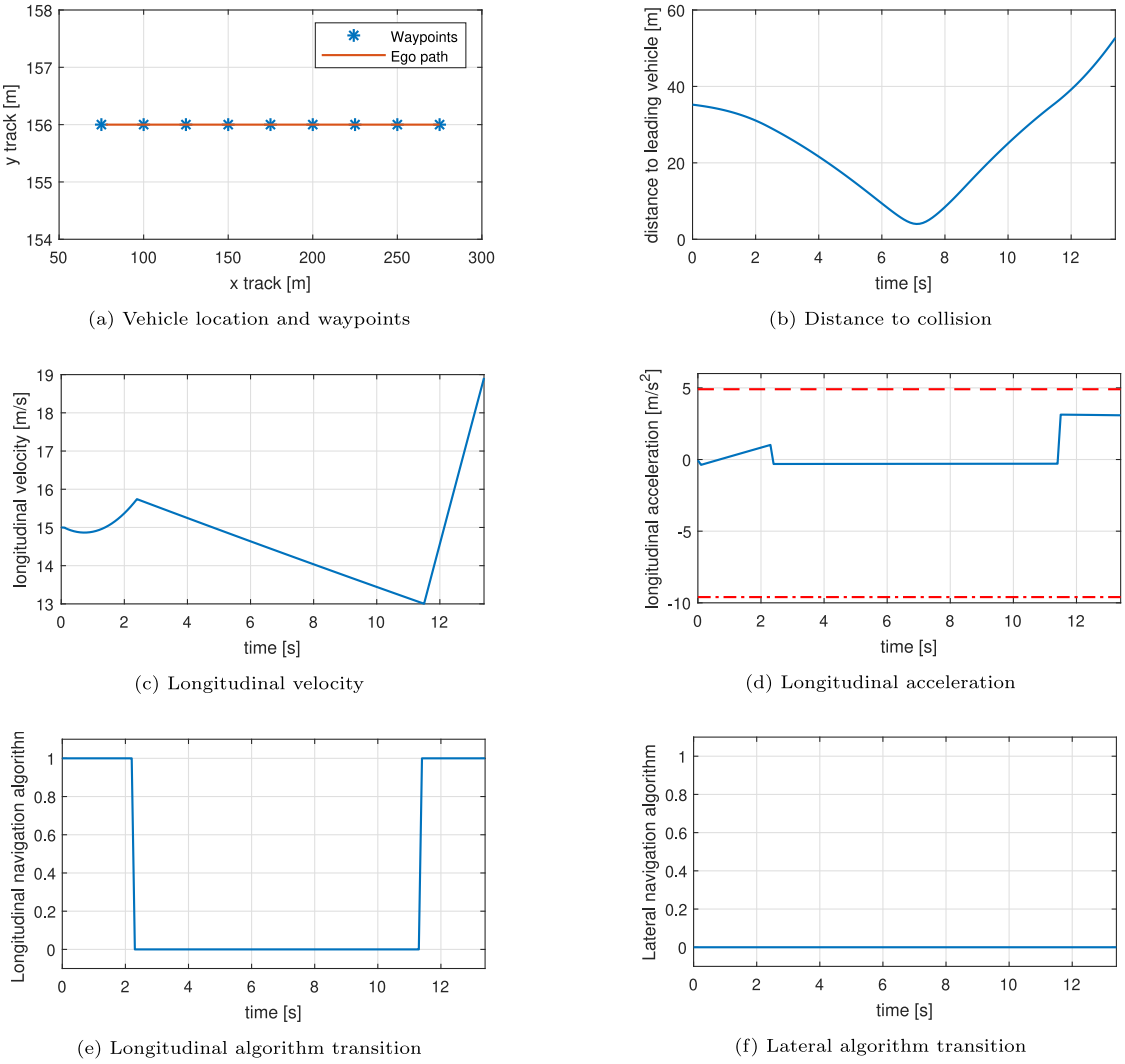
After the coupled RL-MPC agent is tested in modified conditions of the training scenario, its generalization to other scenarios is validated. Three extra scenarios were developed: a single lane straight and a double lane crossroad. The straight road represents a simpler and the crossroad a more complex environment. The metrics are the same as the ones presented in Section 6.3.1.

#### 6.4.1. Single-lane straight road

This scenario is a subset of the original T-intersection environment. Only zones $Z_A, Z'_A, Z_B$ and $B'$ are valid. The ego vehicle starts this simulation in zone $Z_B$. Its initial pose is set to $[X, Y, \psi] = [25\text{ m}, 144\text{ m}, 0\text{ deg}]$, and $\dot{x} = 15$ m/s (Fig. 12). Its path is set to $Z_B \rightarrow Z'_A$. Two extra vehicles are spawned ahead and behind it with a gap of 35 m and 45 m, respectively. Two extra vehicles are evenly distributed in zone $Z_B$ with initial speeds of 7 m/s. The complexity of this path is very low. Therefore, two jaywalkers are added. They are spawned such that they can affect indirectly the trajectory of the ego vehicle. Initially the ego vehicle slows down to keep a safe distance with the leading vehicle. The first jaywalker appears in front of the leading vehicle at $t \approx 10$ s, which causes a sudden brake (Figs. 12(c) and 12(d)). The leading vehicle slows down again at $t \approx 22$ s because of the second jaywalker. Consequently, the speed is decreased. Finally, The ego vehicle continues with its navigation commands until it reaches its goal. The ego vehicle deviates from its original path to compensate the behavior of other elements in the environment. The cross-track error is compensated at the end of the navigation. The fail-operational module takes control of the ego vehicle when unexpected situations are faced, and the main navigation system does not respond as expected.

#### 6.4.2. Single-lane crossroad

This scenario is an extension of the original T-intersection environment. The road that includes zones $Z_C$ and $Z'_C$ are extended such that zones $Z_D$ and $Z'_D$ are created. The ego vehicle starts this simulation in zone $Z_C$. Its initial pose is set to $[X, Y, \psi] = [156\text{ m}, 25\text{ m}, 90\text{ deg}]$, and $\dot{x} = 15$ m/s (Fig. 13(a)). Its path is set to $Z_C \rightarrow Z'_D$. Two extra vehicles are spawned in front of and




(a) Vehicle location and waypoints

(b) Distance to collision

(c) Longitudinal velocity

(d) Longitudinal acceleration

(e) Longitudinal algorithm transition

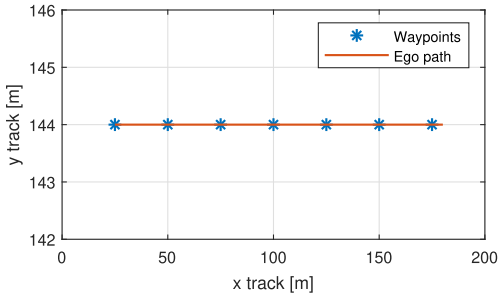
(f) Lateral algorithm transition

**Fig. 11.** Performance metrics RL-MPC coupled navigation for test 3.

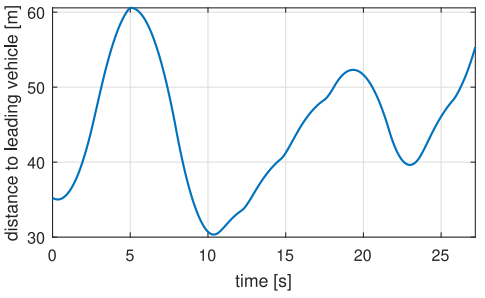
behind it with a gap of 15 m and 25 m respectively. Six extra vehicles are evenly distributed in zones $Z_B, Z_C$ and $Z_D$ with initial speeds of 10 m/s. The complexity of this path is very high. Four jaywalkers are added. They are spawned such that they can directly or indirectly affect the trajectory of the ego vehicle. The first jaywalker appears in front of the leading vehicle at $t \approx 2$ s, which causes a sudden brake as the distance to collision is reduced (Fig. 13(b)). Changes in speed and acceleration can also be observed in Figs. 13(c) and 13(d). The ego vehicle slows down to adjust to the speed of the leading vehicle as it approaches the intersection. The ego vehicle carefully approaches the intersection $t \approx [14, 20]$ s, giving enough room for crossing vehicles. A pedestrian uses *Crosswalk 1* at $t \approx 17$ s causing the leading vehicle to yield. This disturbance can be spotted in the velocity and acceleration plots. It causes the ego vehicle to apply the brake and to navigate at low speeds. Simultaneously, another vehicle coming from area $Z_D$ enters the intersection, compromising the integrity of the agent. A safety distance of at least $t = 4$ s. After it has merged to area $Z'_D$, the ego vehicle continues navigating and maintaining a valid minimum distances to collision until it reaches its goal. The fail-operational module actively took control of the ego vehicle by keeping a safe distance with the surrounding vehicles and pedestrian in a complex environment.

### 6.4.3. Multiple-lane crossroad

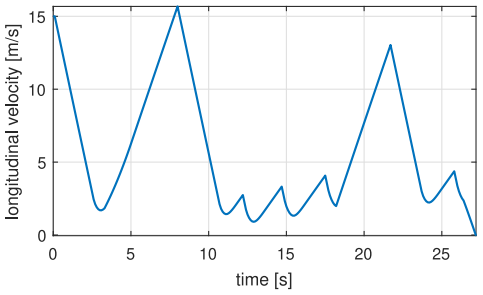
The agent is tested on a multiple lane scenario extended from the one presented in Section 6.4.2. Zones $Z_A, Z'_B, Z_C$ and $Z'_D$ are extended as the topology of the roads becomes two-lane, tow-direction, passing. The ego vehicle starts this simulation in zone $Z_C$. Its initial pose is set to $[X, Y, \psi] = [156\text{ m}, 25\text{ m}, 90\text{ deg}]$, and $\dot{x} = 5$ m/s (Fig. 14(a)). Its path is set to $Z_C \rightarrow Z'_A$. Three extra vehicles with the same path are spawned in the same road and randomly distributed on its lanes. Nine extra vehicles are evenly distributed in zones $Z_B, Z_C$ and $Z_D$ with initial speeds of 12 m/s. The complexity of this scenario is very high. Due to the modifications of



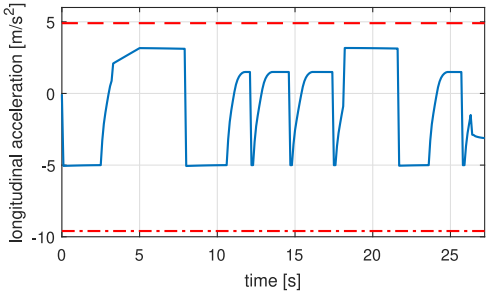

(a) Vehicle location and waypoints




(b) Distance to collision

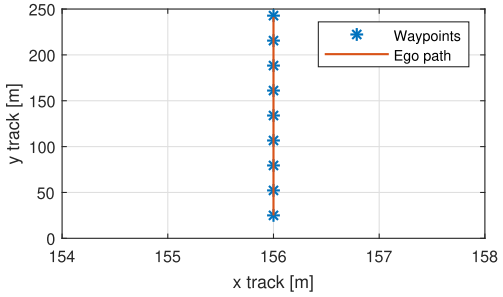



(c) Longitudinal velocity

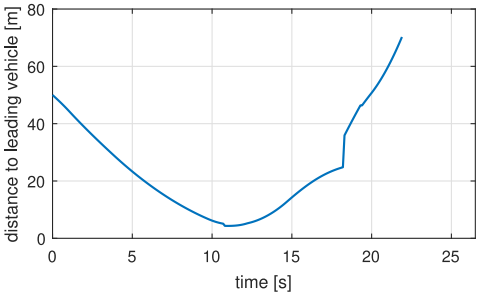



(d) Longitudinal acceleration

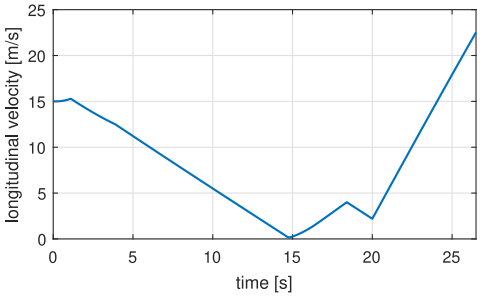
**Fig. 12.** Performance metrics RL-MPC coupled navigation for Section 6.4.1.



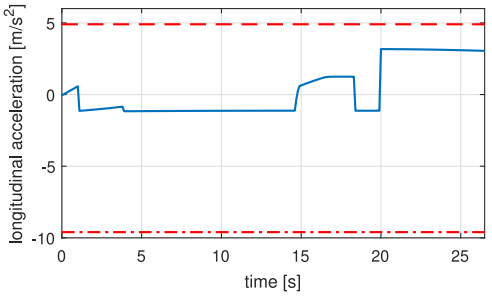

(a) Vehicle location and waypoints




(b) Distance to collision



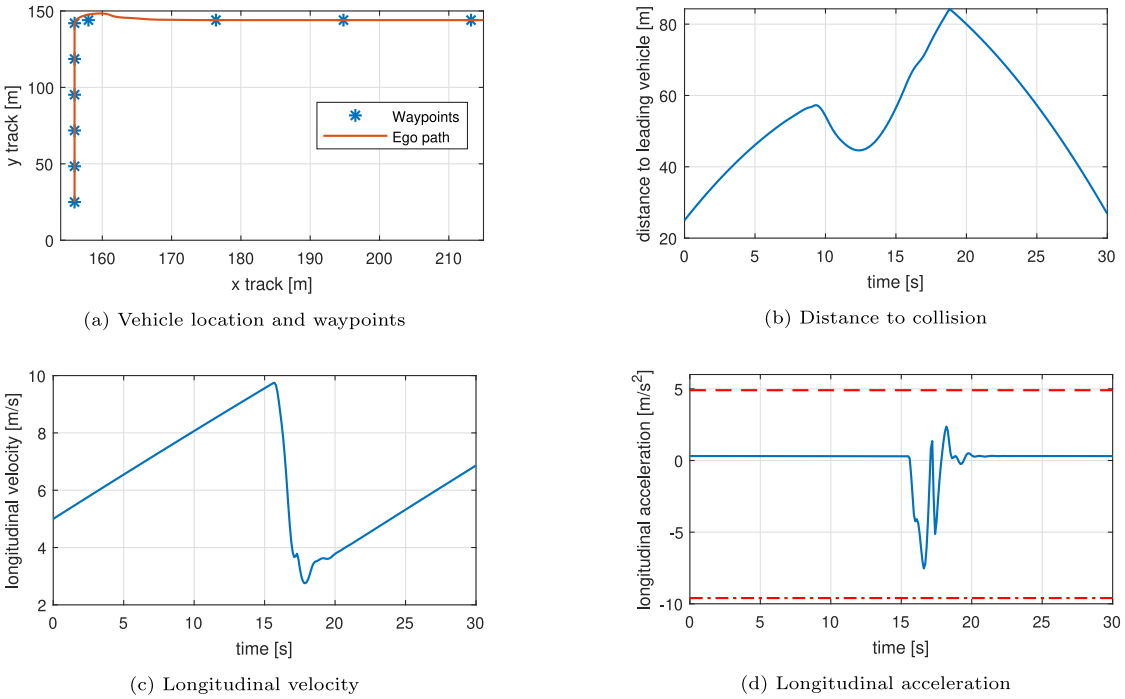

(c) Longitudinal velocity




(d) Longitudinal acceleration

**Fig. 13.** Performance metrics RL-MPC coupled navigation for Section 6.4.2.

the road, the ego vehicle is allowed to change lanes. The task of the vehicle is to reach its goal in the shortest time possible. The TD3-controlled vehicle starts driving forward in the slow-speed lane as it has to turn right. It increases it speed. At $t \approx 12$ s it finds the first vehicle that is heading with the same direction but with a lower speed. The distance to collision is reduced (Fig. 14(b)). However this does not affect the behavior of the vehicle. At this point the vehicle must turn right. This maneuver can be found




(a) Vehicle location and waypoints

(b) Distance to collision

(c) Longitudinal velocity

(d) Longitudinal acceleration

**Fig. 14.** Performance metrics RL-MPC coupled navigation for Section 6.4.3.

in the $t \approx [16, 20]$ s interval. This event can be clearly appreciated in Figs. 14(c) and 14(d). The vehicle briefly uses the adjacent lane while turning. A conservative navigation strategy is kept until the ego vehicle merges into the destination road. The vehicle increases its speed to its destination. This behavior continues until it reaches the goal.

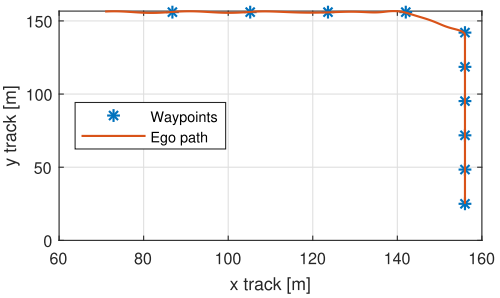
### *6.4.4. Multiple-lane crossroad with slopes*

An extension of the street layout presented in Section 6.4.3 is used to test the performance of the vehicle on sloped roads. The ego vehicle starts this simulation in zone $Z_C$. Its initial pose is set to $[X, Y, Z, \psi] = [156\ \text{m}, 25\ \text{m}, -10\ \text{m}, 90\ \text{deg}]$, and $\dot{x} = 15$ m/s (Fig. 15(a)). Its path is set to $Z_C \rightarrow Z'_B$. Extra vehicles are spawned as in the previous layout. The vehicle starts increasing its speed due to the wide gap between the leading vehicle and itself (Fig. 15(b)). When the safety distance is reduced to a minimum, the vehicle brakes and slowly navigates to the intersection. It then finds a gap to merge to its target road (Figs. 15(c) and 15(d)). Once it is on the destination road, it slowly approaches to its goal as there is a vehicle in front of it. The $Z$-component of the pose of the vehicle is shown in Fig. 15(e). The fail-operational module overrides the navigation whenever the distance to collision reaches hazardous levels. It is responsible for the navigation approach at the intersection and the road merging.

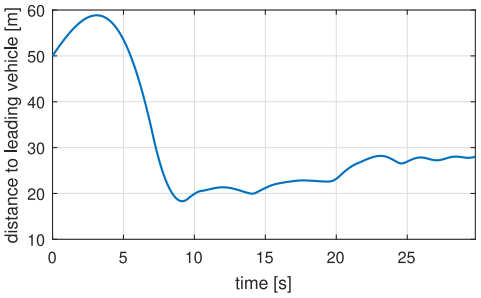
## 7. Conclusions and future research

This work develops a coupled reinforcement learning and model predictive control methods for autonomous vehicles to navigate unsignalized intersections. In a virtual environment, internal and external data are acquired with an odometer and a LIDAR. These data are then fed into a perception system that provides location and behavior information to the agent. A dynamic model of the ego vehicle allows for a more realistic representation of the plant. Furthermore, continuous states and control actions are proposed. The ego vehicle is enhanced with a path planner and lateral controller. An reinforcement learning agent controls the longitudinal acceleration, and selects the navigation path of an ego vehicle. The proposed reward function integrates a combination of nonlinear functions that account for safety, efficiency, and comfort. The agent is trained with the TD3 algorithm, which grants suitable stability and robustness during training. The ego vehicle is trained in a T-intersection environment. Reinforcement learning- and model predictive control-based navigation systems run simultaneously. A safety-enhanced module uses environmental information to select the system that controls the vehicle.

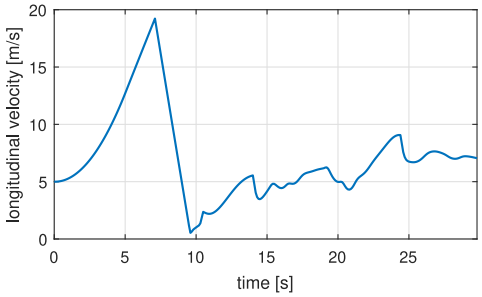
The model is trained within a T-intersection road. It is further tested and validated in other environments including sloped single- or multiple-lane straight road and a crossroad, both achieving satisfactory performance in terms of vehicle integrity and time to



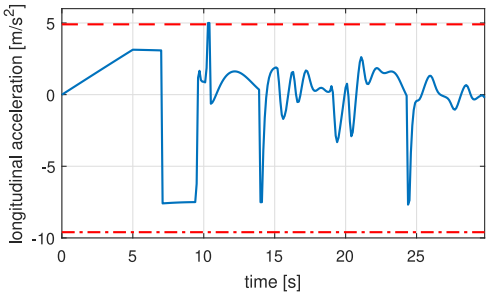

(a) Vehicle location and waypoints



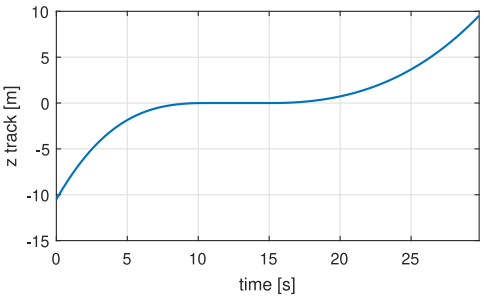

(b) Distance to collision




(c) Longitudinal velocity




(d) Longitudinal acceleration




(e) Vehicle position in the Z axis

**Fig. 15.** Performance metrics RL-MPC coupled navigation for Section 6.4.4.

complete its navigation task. Thus, the achieved architecture enables the possibility of migrating the model to different physical platforms.

The developed work implements a safety-enhanced approach to reinforcement learning-based navigation systems. The specifications of the employed sensors, the vehicle dynamics model, the soft and hard safety constraints, the model predictive controller, and the reward design add more realism to the simulations. These characteristics make this approach more feasible to transfer into a physical platform. The safety module proposed in this work prevents an uncontrolled and rapid switching behavior between algorithms. This module also ensures the navigation of the vehicle to be collision-hazard free.

Navigation of AVs at urban intersections is a relatively understudied field compared to that along highways. We show that the proposed design benchmark can help future researchers explore the extensibility related to reward functions, navigation scenarios, and sensing information. Therefore, there are many open research questions along these lines:

1. The ego vehicle will be enhanced with visual and inertial sensors to acquire further information of the internal and external states. Sensor fusion techniques will be explored to minimize the uncertainty of the environment.
2. Alternative architecture configurations will be evaluated by replacing local path planning and lateral control subsystems. Configuration space, probabilistic sampling or artificial forces can be used as path planners or safety evaluation nodes. Potential controllers can be kinematic or predictive subsystems.

**Table B.6**
Actor network layout properties.

| Layer name | Type | Size | Characteristics |
|---|---|---|---|
| State | Feature input | 544 | |
| actorFC1 | Fully connected | 50 | Weights: 50 × 544, Bias: 50 × 1 |
| actorReLU1 | ReLU activation | 50 | |
| actorFC2 | Fully connected | 50 | Weights: 50 × 50, Bias: 50 × 1 |
| actorReLU2 | ReLU activation | 50 | |
| actorFC3 | Fully connected | 1 | Weights: 1 × 50, Bias: 1 × 1 |
| Action | Hyperbolic tangent | 1 | |

**Table B.7**
Critic network layout properties.

| Layer name | Layer type | Size | Characteristics |
|---|---|---|---|
| Action | Feature input | 1 | |
| CriticAction | Fully connected | 25 | Weights: 25 × 1, Bias: 25 × 1 |
| State | Feature input | 544 | |
| CriticStateFC1 | Fully connected | 50 | Weights: 50 × 544, Bias: 50 × 1 |
| CriticReLU1 | ReLU activation | 50 | |
| CriticStateFC2 | Fully connected | 25 | Weights: 25 × 50, Bias: 25 × 1 |
| add | Addition | 25 | |
| CriticCommonReLU2 | ReLU activation | 25 | |
| CriticOutput | Fully connected | 1 | Weights: 1 × 25, Bias: 1 × 1 |

**Table B.8**
Agent hyperparameters.

| Hyperparameter | Critic network | Actor network |
|---|---|---|
| Learn rate | 0.001 | 0.0001 |
| Regularization factor | 0.0001 | 0.0001 |
| Gradient threshold | 1 | 1 |
| Sample time | 0.1 s | |
| Target smooth factor | 0.001 | |
| Discount factor | 0.999 | |
| Minibatch size | 128 | |
| Experience buffer length | $10^6$ | |

3. Background vehicles will be enhanced with a wide range of reactive and navigation behaviors to better help understand the interaction between vehicles in various navigation scenarios.
4. A multi-agent RL framework will be developed to train multiple AVs that collaboratively navigate an intersection.

## CRediT authorship contribution statement

**Rolando Bautista-Montesano:** Conceptualization, Formal analysis, Methodology, Visualization, Writing – original draft. **Renato Galluzzi:** Formal analysis, Methodology, Visualization, Writing – original draft. **Kangrui Ruan:** Formal analysis, Methodology, Visualization, Writing – original draft. **Yongjie Fu:** Formal analysis, Visualization. **Xuan Di:** Conceptualization, Methodology, Supervision, Writing – review & editing.

## Acknowledgments

Bautista-Montesano and Galluzzi are supported by CIMB at Tecnologico de Monterrey. Bautista-Montesano is funded by Tecnologico de Monterrey - Grant No. A00996397, and Consejo Nacional de Ciencia y Tecnologia (CONACYT) under the scholarship 679120. Ruan, Fu, and Di are supported by NSF CPS-2038984.

## Appendix A. List of symbols

### Path planning

| | |
|---|---|
| $\mathbf{r}$ | Root point |
| $\mathbf{t}_r$, $\mathbf{n}_r$ | Tangential and normal vectors of $\mathbf{r}$ |
| $\mathbf{t}_z$, $\mathbf{n}_z$ | Tangential and normal vectors of $\mathbf{z}$ |
| $\mathbf{z}$ | Computed trajectory |
| $C$ | Trajectory cost function |
| $d_p$ | Perpendicular offset |
| $g$ | Time-dependent cost parameter |
| $h$ | Final position-dependent cost parameter |
| $J_t$ | Trajectory optimization function |
| $k_j, k_t, k_p$ | Trajectory cost function coefficients |
| $p$ | Position |
| $P_0$ | Initial state vector |
| $P_1$ | Final state vector |
| $p_p$ | Position |
| $s_p$ | Arc length of the centerline |
| $T$ | Time step |
| $t_0$ | Initial time |
| $t_1$ | Final time |

### Vehicle model

| | |
|---|---|
| $\delta$ | Steering angle |
| $\theta_{vf}, \theta_{vr}$ | Velocity angles of the front and rear tires |
| $C_{\alpha f}, C_{\alpha r}$ | Cornering stiffness of the front and rear tires |
| $F_{xf}$ | External longitudinal force on the front axle |
| $F_{yf}$, $F_{yr}$ | External lateral forces on the front and rear axles |
| $I_{zz}$ | Polar moment of inertia with respect to the $z$ axis |
| $L$ | Wheelbase |
| $l_f$, $l_r$ | Distance between the center of gravity and the front and rear axles |
| $m$ | Mass |
| $x$,$y$,$\psi$ | Longitudinal, lateral and yaw DOFs of the vehicle |

### Reinforcement learning

| | |
|---|---|
| $\gamma$ | Discount factor |
| $\mathbb{E}$ | Expectation |
| $\mathcal{B}_R$ | Replay buffer |
| $\mathcal{L}(\theta)$ | The loss function for the Q-value |
| $\nabla_\phi$ | Output gradient |
| $\pi$ | Policy network |
| $\pi'$ | Target network for the policy network |
| $o$ | Observation |
| $p$ | Probability function |
| $Q_{\theta'}$ | Target network for the critic network |
| $Q_\theta$ | Critic network |
| $s$ | State |
| $X, Y$ | World-frame coordinates of the ego vehicle |
| $Z_A, Z_B, Z_C, Z_D$ | Road zones |

### Lateral control

| | |
|---|---|
| $\alpha$ | Angle between the heading and look-ahead distance vectors |
| $\delta$ | Steering angle |
| $\kappa$ | Curvature of a path |
| $\varepsilon_{ct}$ | Cross-track error |
| $K_d$ | Speed gain |
| $l_d$ | Look-ahead distance |
| $R_1$ | Curved path radius |

### Reward

| | |
|---|---|
| $\beta$ | Angle to path |
| $\lambda$ | Road deviation |
| $\Lambda_j$ | Lane center |
| $\mathcal{F}_i$ | Frenét path |
| $\tau_c$ | Dynamic collision threshold distance |
| $a_i$ | Actions |
| $d_c$ | Distance to collision |
| $F_k$ | Input force |
| $F_{max}$ | Maximum input force |
| $K$ | Progress estimation gain |
| $r_d$ | Distance-to-collision reward |
| $r_\beta$ | Angle-to-path reward |
| $r_{\ddot{x}}$ | Longitudinal acceleration reward |
| $r_{\dot{x}}$ | Longitudinal speed reward |
| $r_{\dot{\psi}}$ | Yaw rate reward |
| $r_{C_{fp}}$ | Selected path reward |
| $r_{\epsilon_{ct}}$ | Cross-track error reward |
| $w_i$ | Reward function weights |

### Safety

| | |
|---|---|
| $\eta$ | Extended output vector |
| $\tau_a$ | Longitudinal model time constant |
| $\tilde{A}$ | Extended state matrix |
| $\tilde{B}$ | Extended input matrix |
| $\tilde{C}$ | Extended output matrix |
| $\tilde{x}, \tilde{u}$ | State and input linearization points |
| $\xi$ | Extended state vector |
| $A$ | State matrix Jacobian |
| $B$ | Input matrix Jacobian |
| $e$ | Error |
| $J_s$ | Safety MPC cost function |
| $k$ | Time sample index |
| $N_c$ | Control horizon |
| $N_p$ | Prediction horizon |

## Appendix B. Neural network layout

See Tables B.6–B.8.

## References


Alonso, E., Peter, M., Goumard, D., Romoff, J., 2020. Deep reinforcement learning for navigation in AAA video games. arXiv:2011.04764.

Amer, N.H., Zamzuri, H., Hudha, K., Kadir, Z.A., 2016. Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges. J. Intell. Robot. Syst. 86 (2), 225–254. http://dx.doi.org/10.1007/s10846-016-0442-0.

Baheri, A., Kolmanovsky, I., Girard, A., Tseng, H.E., Filev, D., 2020a. Vision-based autonomous driving: A model learning approach. In: 2020 American Control Conference (ACC). pp. 2520–2525. http://dx.doi.org/10.23919/ACC45564.2020.9147510.

Baheri, A., Nageshrao, S., Tseng, H.E., Kolmanovsky, I., Girard, A., Filev, D., 2020b. Deep reinforcement learning with enhanced safety for autonomous highway driving. In: 2020 IEEE Intelligent Vehicles Symposium (IV). pp. 1550–1555. http://dx.doi.org/10.1109/IV47402.2020.9304744.

Behere, S., Torngren, M., 2015. A functional architecture for autonomous driving. In: 2015 First International Workshop on Automotive Software Architecture (WASA). IEEE, pp. 3–10. http://dx.doi.org/10.1145/2752489.2752491.

Bojarski, M., del Testa, D.W., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K., 2016. End to end learning for self-driving cars. arXiv arXiv:1604.07316.

Borhaug, E., Pettersen, K.Y., 2005. Cross-track control for underactuated autonomous vehicles. In: Proceedings of the 44th IEEE Conference on Decision and Control. pp. 602–608. http://dx.doi.org/10.1109/CDC.2005.1582222.

Boroujeni, Z., Goehring, D., Ulbrich, F., Neumann, D., Rojas, R., 2017. Flexible unit A-star trajectory planning for autonomous vehicles on structured road maps. In: 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES). IEEE, pp. 7–12. http://dx.doi.org/10.1109/ICVES.2017.7991893.

Bouton, M., Nakhaei, A., Fujimura, K., Kochenderfer, M.J., 2019. Safe reinforcement learning with scene decomposition for navigating complex urban environments. In: 2019 IEEE Intelligent Vehicles Symposium (IV). pp. 1469–1476. http://dx.doi.org/10.1109/IVS.2019.8813803.

Chazette, P., Totems, J., Hespel, L., Bailly, J.-S., 2016. 5 - principle and physics of the LiDAR measurement. In: Baghdadi, N., Zribi, M. (Eds.), Optical Remote Sensing of Land Surface. Elsevier, pp. 201–247. http://dx.doi.org/10.1016/B978-1-78548-102-4.50005-3.

Chen, S.-P., Xiong, G.-M., Chen, H.-Y., Negrut, D., 2020. MPC-based path tracking with PID speed control for high-speed autonomous vehicles considering time-optimal travel. J. Cent. South Univ. 27 (12), 3702–3720. http://dx.doi.org/10.1007/s11771-020-4561-1.

Chen, J., Zhao, P., Mei, T., Liang, H., 2013. Lane change path planning based on piecewise bezier curve for autonomous vehicle. In: Proceedings of 2013 IEEE International Conference on Vehicular Electronics and Safety. IEEE, pp. 17–22. http://dx.doi.org/10.1109/ICVES.2013.6619595.

Chowdhri, N., Ferranti, L., Iribarren, F.S., Shyrokau, B., 2021. Integrated nonlinear model predictive control for automated driving. Control Eng. Pract. 106, 104654. http://dx.doi.org/10.1016/j.conengprac.2020.104654.

Chua, K., Calandra, R., McAllister, R., Levine, S., 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. In: NIPS'18, Curran Associates Inc., Red Hook, NY, USA, pp. 4759–4770.

Claussmann, L., Revilloud, M., Glaser, S., Gruyer, D., 2017. A study on al-based approaches for high-level decision making in highway autonomous driving. In: IEEE International Conference on Systems, Man and Cybernetics (SMC). IEEE, pp. 3671–3676. http://dx.doi.org/10.1109/smc.2017.8123203.

Claussmann, L., Revilloud, M., Gruyer, D., Glaser, S., 2020. A review of motion planning for highway autonomous driving. IEEE Trans. Intell. Transp. Syst. 21, 1826–1848. http://dx.doi.org/10.1109/TITS.2019.2913998.

Connell, D., La, H.M., 2017. Dynamic path planning and replanning for mobile robots using rrt. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, pp. 1429–1434. http://dx.doi.org/10.1109/SMC.2017.8122814.

Coulter, R.C., 1992. Implementation of the Pure Pursuit Path Tracking Algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA.

Demarsin, K., Vanderstraeten, D., Volodine, T., Roose, D., 2007. Detection of closed sharp edges in point clouds using normal estimation and graph theory. Comput. Aided Des. 39 (4), 276–283. http://dx.doi.org/10.1016/j.cad.2006.12.005.

Di, X., Shi, R., 2021. A survey on autonomous vehicle control in the era of mixed-autonomy: From physics-based to AI-guided driving policy learning. Transp. Res. C 125, 103008. http://dx.doi.org/10.1016/j.trc.2021.103008.

Dijkstra, E.W., et al., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271.

Ding, W., Wang, W., Zhao, D., 2019. A multi-vehicle trajectories generator to simulate vehicle-to-vehicle encountering scenarios. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 4255–4261. http://dx.doi.org/10.1109/ICRA.2019.8793776.

Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J., 2008. Practical search techniques in path planning for autonomous driving. Ann. Arbor. 1001 (48105), 18–80.

Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T., 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. In: ICML Workshop on Real-Life RL, Vol. 110. Springer Science and Business Media LLC, pp. 2419–2468. http://dx.doi.org/10.1007/s10994-021-05961-4.

Fan, T., Long, P., Liu, W., Pan, J., 2020. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. Int. J. Robot. Res. 39 (7), 856–892. http://dx.doi.org/10.1177/0278364920916531.

Fitzpatrick, K., Brewer, M.A., Turner, S., 2006. Another look at pedestrian walking speed. Transp. Res. Rec. 1982 (1), 21–29. http://dx.doi.org/10.1177/0361198106198200104.

Fraden, J., 2016. Handbook of Modern Sensors. Springer International Publishing.

Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. pp. 1587–1596, arXiv:1802.09477.

Genta, G., 2009. The Automotive Chassis – Vol. 2. Springer International Publishing.

Gromniak, M., Stenzel, J., 2019. Deep reinforcement learning for mobile robot navigation. In: 2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS). pp. 68–73. http://dx.doi.org/10.1109/ACIRS.2019.8935944.

Gros, S., Zanon, M., 2020. Reinforcement learning for mixed-integer problems based on MPC. IFAC-PapersOnLine 53 (2), 5219–5224. http://dx.doi.org/10.1016/j.ifacol.2020.12.1196, URL: https://www.sciencedirect.com/science/article/pii/S2405896320315913. 21st IFAC World Congress.

Gu, Z., Li, Z., Di, X., Shi, R., 2020. An LSTM-based autonomous driving model using a waymo open dataset. Appl. Sci. 10 (6), 2046.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning. PMLR, pp. 1861–1870.

Hwan Jeon, J., Cowlagi, R.V., Peters, S.C., Karaman, S., Frazzoli, E., Tsiotras, P., Iagnemma, K., 2013. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In: 2013 American Control Conference. IEEE, pp. 188–193. http://dx.doi.org/10.1109/ACC.2013.6579835.

Isele, D., Rahimi, R., Cosgun, A., Subramanian, K., Fujimura, K., 2018. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 2034–2039. http://dx.doi.org/10.1109/ICRA.2018.8461233.

Jazar, R.N., 2009. Vehicle Dynamics: Theory and Application. Springer, New York, USA.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al., 2018. Scalable deep reinforcement learning for vision-based robotic manipulation. In: Conference on Robot Learning. PMLR, pp. 651–673, arXiv:1806.1029.

Kamrani, M., Srinivasan, A.R., Chakraborty, S., Khattak, A.J., 2020. Applying Markov decision process to understand driving decisions using basic safety messages data. Transp. Res. C 115, 102642. http://dx.doi.org/10.1016/j.trc.2020.102642.

Karnchanachari, N., de la Iglesia Valls, M., Hoeller, D., Hutter, M., 2020. Practical reinforcement learning for MPC: Learning from sparse objectives in under an hour on a real robot. In: Bayen, A.M., Jadbabaie, A., Pappas, G., Parrilo, P.A., Recht, B., Tomlin, C., Zeilinger, M. (Eds.), Proceedings of the 2nd Conference on Learning for Dynamics and Control. In: Proceedings of Machine Learning Research, vol. 120, PMLR, pp. 211–224, URL: https://proceedings.mlr.press/v120/karnchanachari20a.html.

Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P., 2021. Deep reinforcement learning for autonomous driving: A survey. IEEE Trans. Intell. Transp. Syst. 1–18. http://dx.doi.org/10.1109/TITS.2021.3054625.

Konda, V.R., Tsitsiklis, J.N., 2000. Actor-critic algorithms. In: Advances in Neural Information Processing Systems. Citeseer, pp. 1008–1014.

Kordabad, A.B., Esfahani, H.N., Lekkas, A.M., Gros, S., 2021. Reinforcement learning based on scenario-tree MPC for ASVs. In: 2021 American Control Conference (ACC). pp. 1985–1990. http://dx.doi.org/10.23919/ACC50511.2021.9483100.

Kreidieh, A.R., Wu, C., Bayen, A.M., 2018. Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 1475–1480. http://dx.doi.org/10.1109/ITSC.2018.8569485.

LaValle, S.M., et al., 1998. Rapidly-exploring random trees: A new tool for path planning. TR.

Li, N., Chen, H., Kolmanovsky, I., Girard, A., 2017. An explicit decision tree approach for automated driving. In: Proceedings of the ASME 2017. American Society of Mechanical Engineers, pp. 1–11. http://dx.doi.org/10.1115/dscc2017-5099.

Li, N., Girard, A., Kolmanovsky, I., 2019. Stochastic predictive control for partially observable Markov decision processes with time-joint chance constraints and application to autonomous vehicle control. J. Dyn. Syst. Meas. Control 141 (7), http://dx.doi.org/10.1115/1.4043115, 071007.

Li, Y., Li, N., Tseng, H.E., Girard, A., Filev, D., Kolmanovsky, I., 2021. Safe reinforcement learning using robust action governor. arXiv:2102.10643.

Li, N., Oyler, D.W., Zhang, M., Yildiz, Y., Kolmanovsky, I., Girard, A.R., 2018. Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. IEEE Trans. Control Syst. Technol. 26 (5), 1782–1797. http://dx.doi.org/10.1109/TCST.2017.2723574.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lima, P.F., Trincavelli, M., Mårtensson, J., Wahlberg, B., 2015. Clothoid-based speed profiler and control for autonomous driving. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems. IEEE, pp. 2194–2199. http://dx.doi.org/10.1109/ITSC.2015.354.

Lin, F., Zhang, Y., Zhao, Y., Yin, G., Zhang, H., Wang, K., 2019. Trajectory tracking of autonomous vehicle with the fusion of DYC and longitudinal–lateral control. Chin. J. Mech. Eng. 32 (1), http://dx.doi.org/10.1186/s10033-019-0327-9.

Liu, F., Chen, C., Li, Z., Guan, Z.H., Wang, H.O., 2020. Research on path planning of robot based on deep reinforcement learning. In: 2020 39th Chinese Control Conference (CCC). pp. 3730–3734. http://dx.doi.org/10.23919/CCC50068.2020.9188890.

Ma, X., Li, J., Kochenderfer, M.J., Isele, D., Fujimura, K., 2021. Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships. arXiv:2011.04251.

Marchesini, E., Farinelli, A., 2020. Discrete deep reinforcement learning for mapless navigation. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 10688–10694. http://dx.doi.org/10.1109/ICRA40945.2020.9196739.

Matignon, L., Laurent, G.J., Le Fort-Piat, N., 2006. Reward function and initial values: Better choices for accelerated goal-directed reinforcement learning. In: International Conference on Artificial Neural Networks. Springer, pp. 840–849. http://dx.doi.org/10.1007/11840817_87.

McAllister, R., Gal, Y., Kendall, A., van der Wilk, M., Shah, A., Cipolla, R., Weller, A., 2017. Concrete problems for autonomous vehicle safety: Advantages of Bayesian deep learning. In: Proceedings. International Joint Conferences on Artificial Intelligence Organization, pp. 4745–4753. http://dx.doi.org/10.24963/ijcai.2017/661.

Miloradović, D., Glisovic, J., Stojanovic, N., Grujic, I., 2019. Simulation of vehicle's lateral dynamics using nonlinear model with real inputs. In: IOP Conference Series: Materials Science and Engineering, Vol. 659. IOP Publishing, 012060. http://dx.doi.org/10.1088/1757-899X/659/1/012060.

Morinelly, J.E., Ydstie, B.E., 2016. Dual MPC with reinforcement learning. IFAC-PapersOnLine 49 (7), 266–271. http://dx.doi.org/10.1016/j.ifacol.2016.07.276, URL: https://www.sciencedirect.com/science/article/pii/S2405896316304839. 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016.

Nam, H., Choi, W., Ahn, C., 2019. Model predictive control for evasive steering of an autonomous vehicle. Int. J. Automot. Technol. 20 (5), 1033–1042. http://dx.doi.org/10.1007/s12239-019-0097-5.

Rajamani, R., 2006. Vehicle Dynamics and Control. Springer, New York, USA.

Russell, S., Norvig, P., 2009. Artificial Intelligence: A Modern Approach, third ed. Prentice Hall Press, USA.

Salvucci, D.D., 2006. Modeling driver behavior in a cognitive architecture. Hum. Factors 48 (2), 362–380. http://dx.doi.org/10.1518/001872006777724417.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Secretary, I.C., 2019. Passenger Cars – Braking in a Turn – Open-Loop Test Method. International Organization for Standardization, Geneva, CH.

Shou, Z., Chen, X., Fu, Y., Di, X., 2022. Multi-agent reinforcement learning for markov routing games: a new modeling paradigm for dynamic traffic assignment. Transp. Res. C 137 (103560), http://dx.doi.org/10.1016/j.trc.2022.103560.

Shou, Z., Di, X., 2020. Reward design for driver repositioning using multi-agent reinforcement learning. Transp. Res. C 119 (102738), http://dx.doi.org/10.1016/J.TRC.2020.102738.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.

Takahashi, A., Hongo, T., Ninomiya, Y., Sugimoto, G., 1989. Local path planning and motion control for agv in positioning. In: Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems. '(IROS'89)' the Autonomous Mobile Robots and its Applications. IEEE, pp. 392–397. http://dx.doi.org/10.1109/IROS.1989.637936.

Thrun, S., Burgard, W., Fox, D., 2006. Probabilistic Robotics. MIT Press, Cambridge, Mass..

Uber, 2020. Forbes. forbes.com/sites/bizcarson/2017/12/22/ubers-self-driving-cars-2-million-miles [Online; accessed 12.31.2020].

Vanholme, B., Gruyer, D., Lusetti, B., Glaser, S., Mammar, S., 2012. Highly automated driving on highways based on legal safety. IEEE Trans. Intell. Transp. Syst. 14 (1), 333–347. http://dx.doi.org/10.1109/TITS.2012.2225104.

Vinitsky, E., Kreidieh, A., Flem, L.L., Kheterpal, N., Jang, K., Wu, C., Wu, F., Liaw, R., Liang, E., Bayen, A.M., 2018. Benchmarks for reinforcement learning in mixed-autonomy traffic. In: Billard, A., Dragan, A., Peters, J., Morimoto, J. (Eds.), Proceedings of the 2nd Conference on Robot Learning. In: Proceedings of Machine Learning Research, vol. 87, PMLR, pp. 399–409.

Von Hundelshausen, F., Himmelsbach, M., Hecker, F., Mueller, A., Wuensche, H.-J., 2008. Driving with tentacles: Integral structures for sensing and motion. J. Field Robotics 25 (9), 640–673. http://dx.doi.org/10.1007/978-3-642-03991-1_10.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., Ba, J., 2019. Benchmarking model-based reinforcement learning. arXiv preprint arXiv:1907.02057.

Wang, J., Elfwing, S., Uchibe, E., 2021. Modular deep reinforcement learning from reward and punishment for robot navigation. Neural Netw. 135, 115–126. http://dx.doi.org/10.1016/j.neunet.2020.12.001.

Waymo, 2018. The world's longest and toughest ongoing driving test. medium.com/waymo/the-worlds-longest-and-toughest-ongoing-driving-test-44464867865b [Online; accessed 12.31.2020].

Werling, M., Ziegler, J., Kammel, S., Thrun, S., 2010. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In: 2010 IEEE International Conference on Robotics and Automation. IEEE, pp. 987–993. http://dx.doi.org/10.1109/ROBOT.2010.5509799.

Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J.M., Boots, B., Theodorou, E.A., 2017. Information theoretic MPC for model-based reinforcement learning. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 1714–1721. http://dx.doi.org/10.1109/ICRA.2017.7989202.

Wu, X., Chen, H., Chen, C., Zhong, M., Xie, S., Guo, Y., Fujita, H., 2020. The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method. Knowl.-Based Syst. 196, 105201. http://dx.doi.org/10.1016/j.knosys.2019.105201.

Wu, C., Kreidieh, A., Parvate, K., Vinitsky, E., Bayen, A.M., 2017a. Flow: Architecture and benchmarking for reinforcement learning in traffic control. arXiv preprint arXiv:1710.05465.

Wu, C., Parvate, K., Kheterpal, N., Dickstein, L., Mehta, A., Vinitsky, E., Bayen, A.M., 2017b. Framework for control and deep reinforcement learning in traffic. In: Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on. IEEE, pp. 1–8. http://dx.doi.org/10.1109/ITSC.2017.8317694.

Xu, P., Dherbomez, G., Hery, E., Abidli, A., Bonnifait, P., 2018. System architecture of a driverless electric car in the grand cooperative driving challenge. IEEE Intell. Transp. Syst. Mag. 10 (1), 47–59. http://dx.doi.org/10.1109/MITS.2017.2776135.

Yang, L., Yue, M., Ma, T., 2019. Path following predictive control for autonomous vehicles subject to uncertain tire-ground adhesion and varied road curvature. Int. J. Control Autom. Syst. 17 (1), 193–202. http://dx.doi.org/10.1007/s12555-017-0457-8.

Yaqoob, I., Khan, L.U., Kazmi, S.M.A., Imran, M., Guizani, N., Hong, C.S., 2020. Autonomous driving cars in smart cities: Recent advances, requirements, and challenges. IEEE Netw. 34 (1), 174–181. http://dx.doi.org/10.1109/MNET.2019.1900120.

Yasuda, Y.D.V., Martins, L.E.G., Cappabianco, F.A.M., 2020. Autonomous visual navigation for mobile robots: A systematic literature review. ACM Comput. Surv. 53 (1), http://dx.doi.org/10.1145/3368961.

Zanon, M., Gros, S., 2021. Safe reinforcement learning using robust MPC. IEEE Trans. Automat. Control 66 (8), 3638–3652. http://dx.doi.org/10.1109/TAC.2020.3024161.

Zanon, M., Gros, S., Bemporad, A., 2019. Practical reinforcement learning of stabilizing economic MPC. In: 2019 18th European Control Conference (ECC). pp. 2258–2263. http://dx.doi.org/10.23919/ECC.2019.8795816.

Zeng, F., Wang, C., Ge, S.S., 2020. A survey on visual navigation for artificial agents with deep reinforcement learning. IEEE Access 8, 135426–135442. http://dx.doi.org/10.1109/ACCESS.2020.3011438.

Zhang, H., Yu, T., 2020. Taxonomy of reinforcement learning algorithms. In: Deep Reinforcement Learning. Springer Singapore, pp. 125–133. http://dx.doi.org/10.1007/978-981-15-4095-0_3.

Ziegler, J., Stiller, C., 2009. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 1879–1884. http://dx.doi.org/10.1109/IROS.2009.5354448.